BD Connection How do you do DB test/verification

Technical aspect:

- Understand how to run test manually (data flow logic)

1. Understand what is the DB type

2. Include the drivers in you POM.XML

3. Load and register the driver

4. Create a connection with given creds

5. Create a statement

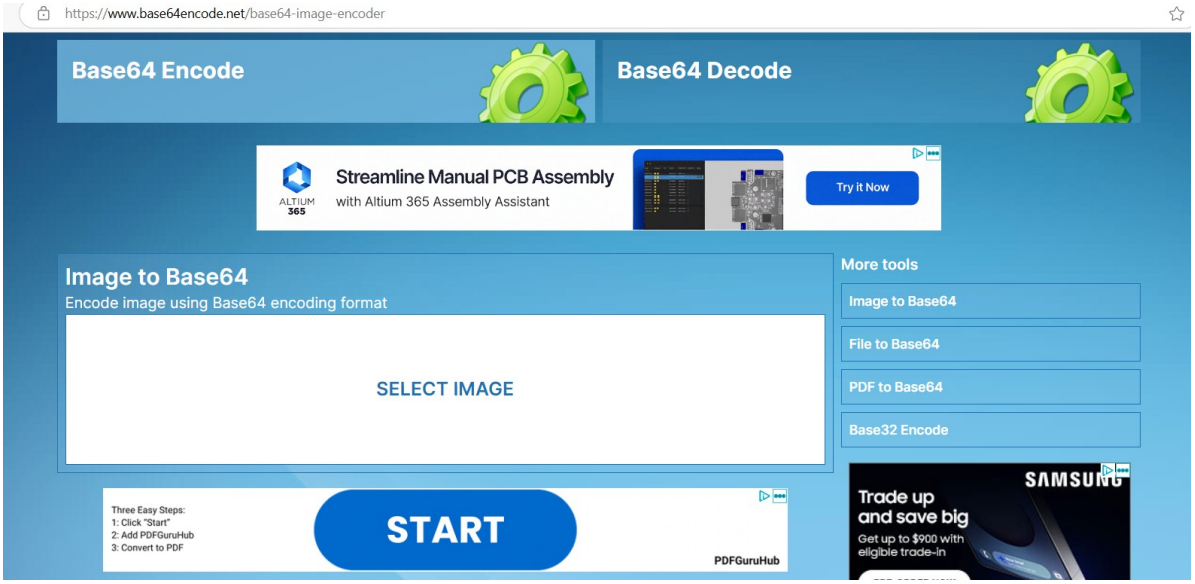
6. Execute a query

7. Get a response and process it

8. Close all the connections at the end

Encode image using Base64 encoding format using this website:

[Image to Base64 Encode - Online Tool]



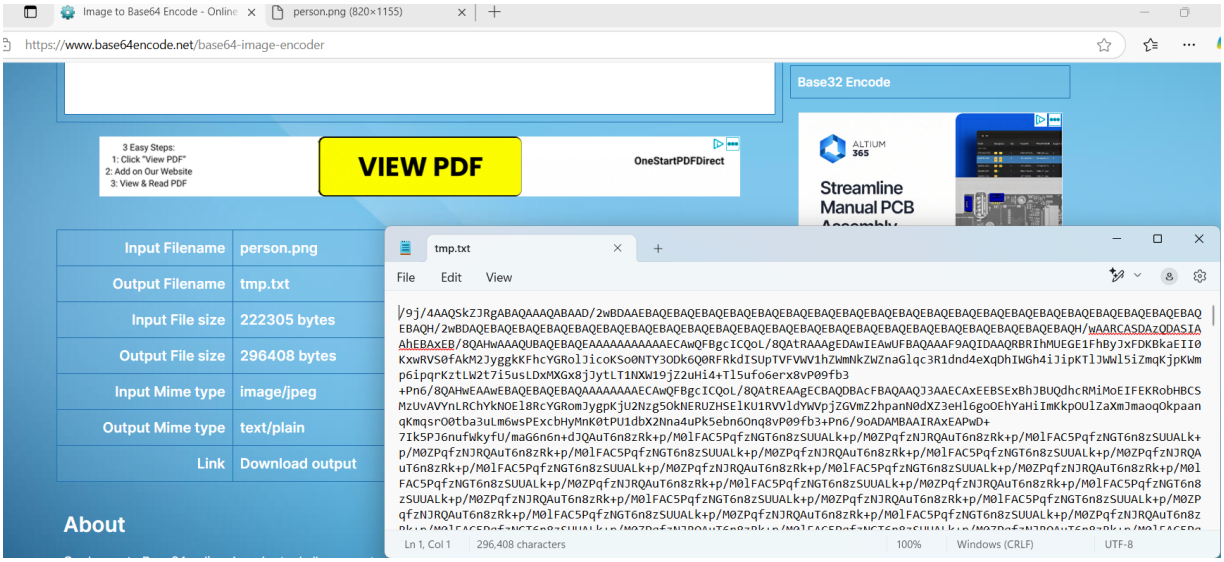
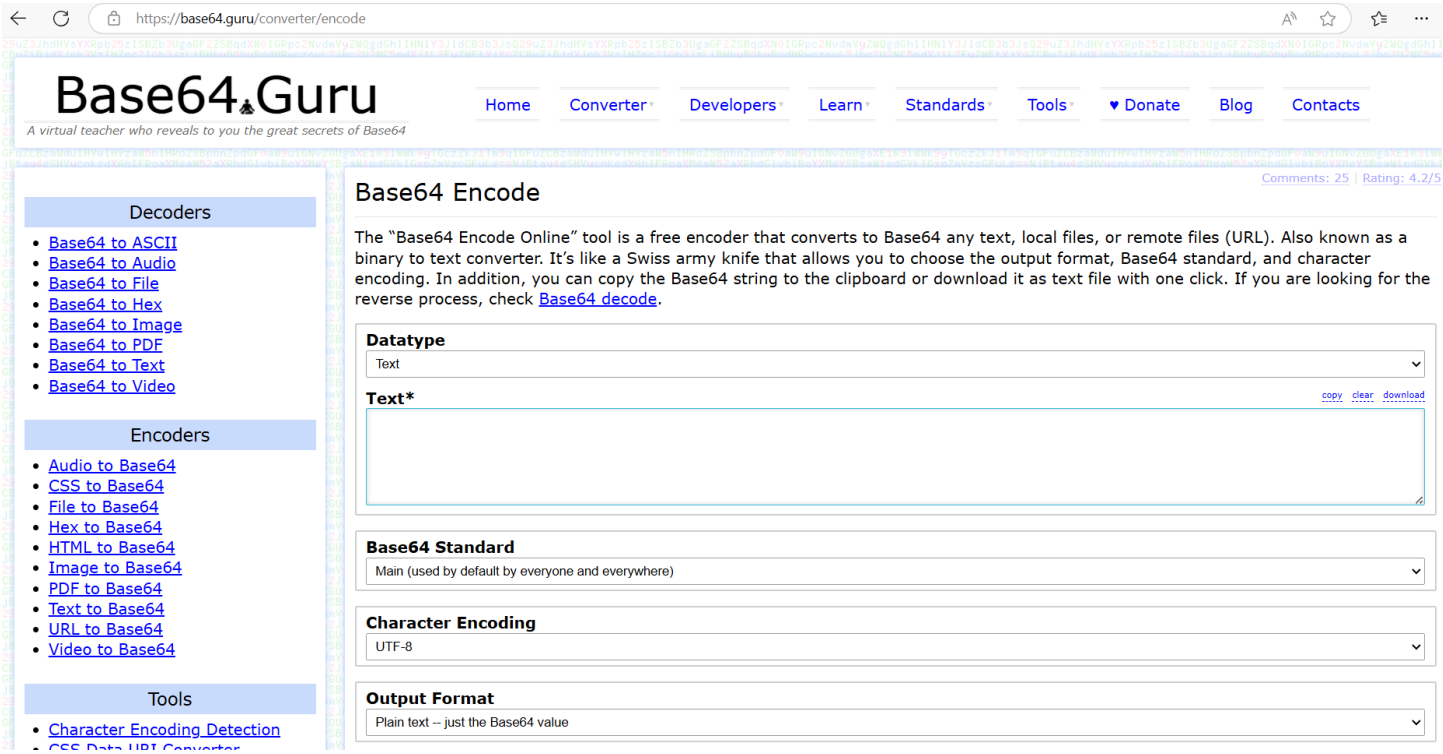


Image to Base64 Encode - Online
person.png (820×1155)
https://www.base64encode.net/base64-image-encoder
Base32 Encode
3 Easy Steps:
1: Click "View PDF"
2: Add on Our Website
3: View & Read PDF
VIEW PDF
OneStartPDFDirect
Input Filename
person.png
Output Filename
tmp.txt
Input File size
222305 bytes
Output File size
296408 bytes
Input Mime type
image/jpeg
Output Mime type
text/plain
Link
Download output
About
tmp.txt
File
Edit
View
/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAEBAQEBAQEBAQ



Additional encoders

There are also some additional Base64 encoders that specialize in the encoding of a particular data type:

- **Audio**
- **CSS**
- **File**
- **Hex**
- **HTML**
- **Image**
- **PDF**
- **Text**
- **URL**
- **Video**

Decoders

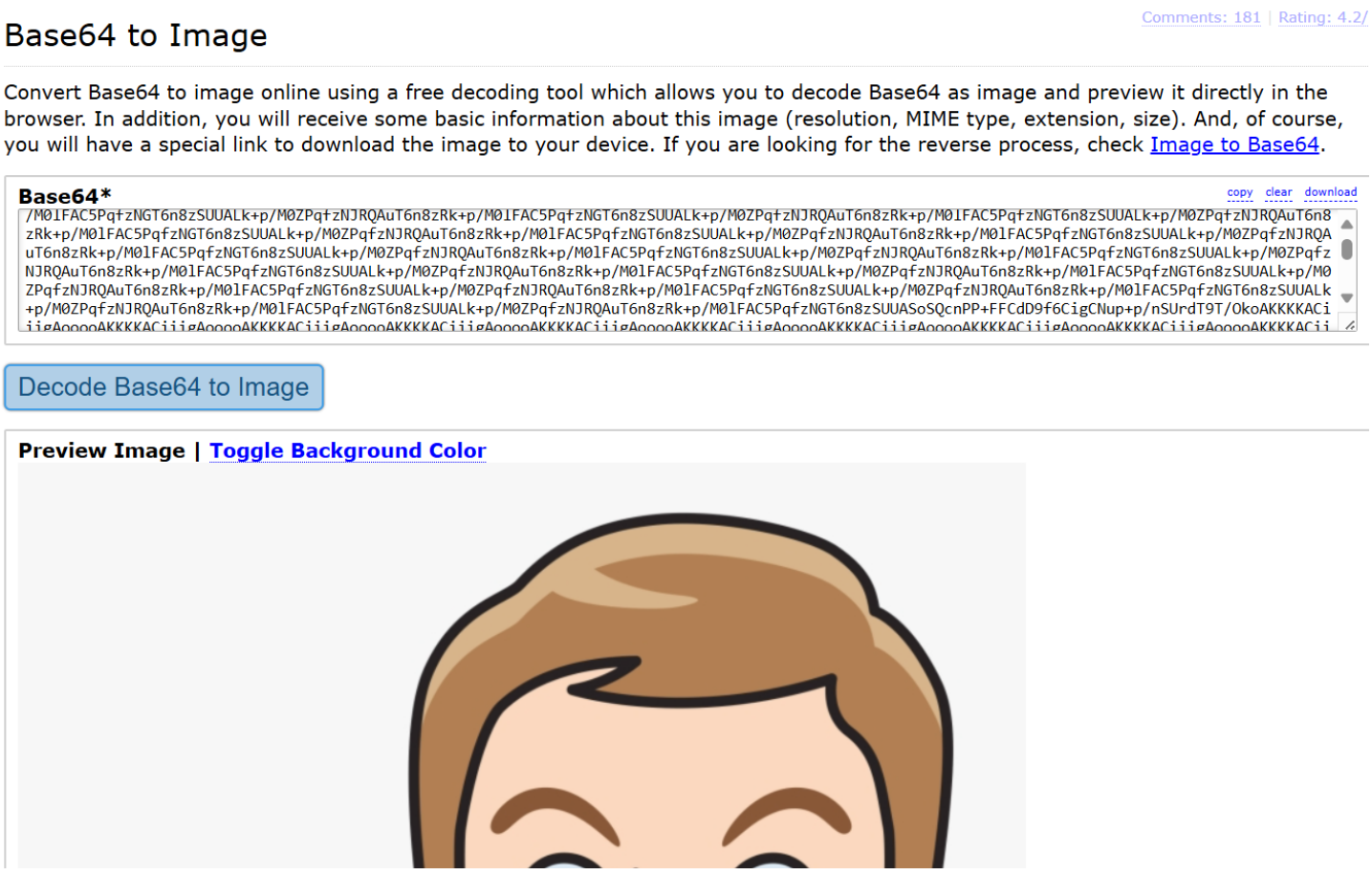
- Base64 to ASCII
- Base64 to Audio
- Base64 to File
- Base64 to Hex
- Base64 to Image

- Base64 to PDF
- Base64 to Text
- Base64 to Video

Convert Base64 to image online using a free decoding tool which allows you to decode Base64 as image and preview it directly in the browser. In addition, you will receive some basic information about this image (resolution, MIME type, extension, size).

Base64 (very long text file) of the image:

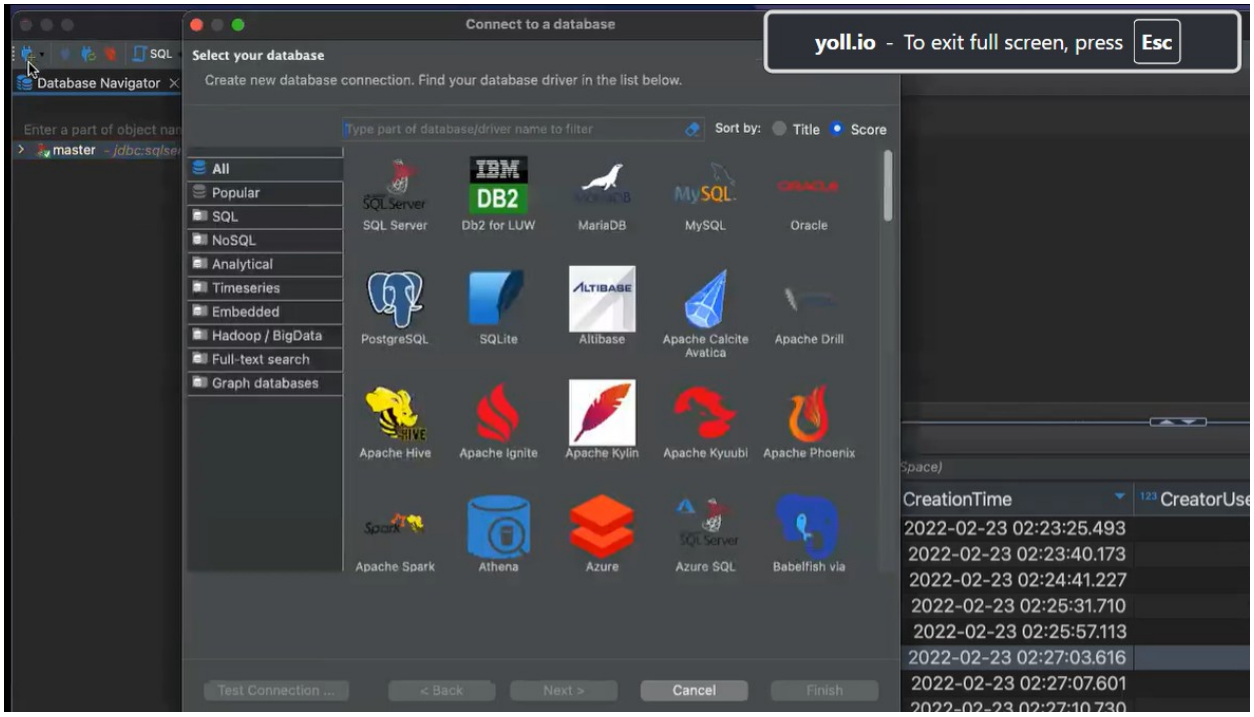
/9j/4AAQSkZJRgABAQAAAQABAAD/
2wBDAAEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQH/
2wBDAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQH/
wAARCASDAzQDASIAAhEBAxEB/8QAHwAAAQUBAQEBAQEAAAAAAAAAAAECAwQFBgcICQoL/
8QAtRAAAgEDAwIEAwUFBAQAAAF9AQIDAAQRBRIhMUEGE1FhByJxFDKBkaEII0KxwRVS0fAkM2JyggkKFhcYGRolJicoKSo0NTY3ODk6Q0RFRkdISUpTVFVWV1h
ZWmNkZWZnaGlqc3R1dnd4eXqDhIWGh4iJipKTlJWWl5iZmqKjpKWmp6ipqrKztLW2t7i5usLDxMXGx8jJytLT1NXW19jZ2uHi4+Tl5ufo6erx8vP09fb3+Pn6/8QAH
wEAAwEBAQEBAQEBAQAAAAAAAAECAwQFBgcICQoL/8QAtREAAgECBAQDBAc



File Info

- Resolution: 820×1155
- MIME type: image/jpeg
- Extension: jpg
- Size: 217.09 KB
- Download: image.jpg
- Channels: 3
- Bit depth: 8

Establishing DataBase and Intellij Connexion with: dbeaver



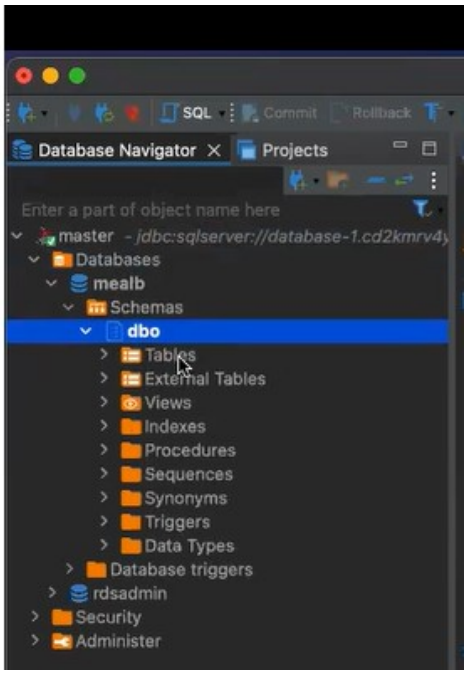
1.Choose Microsoft SQL Server.

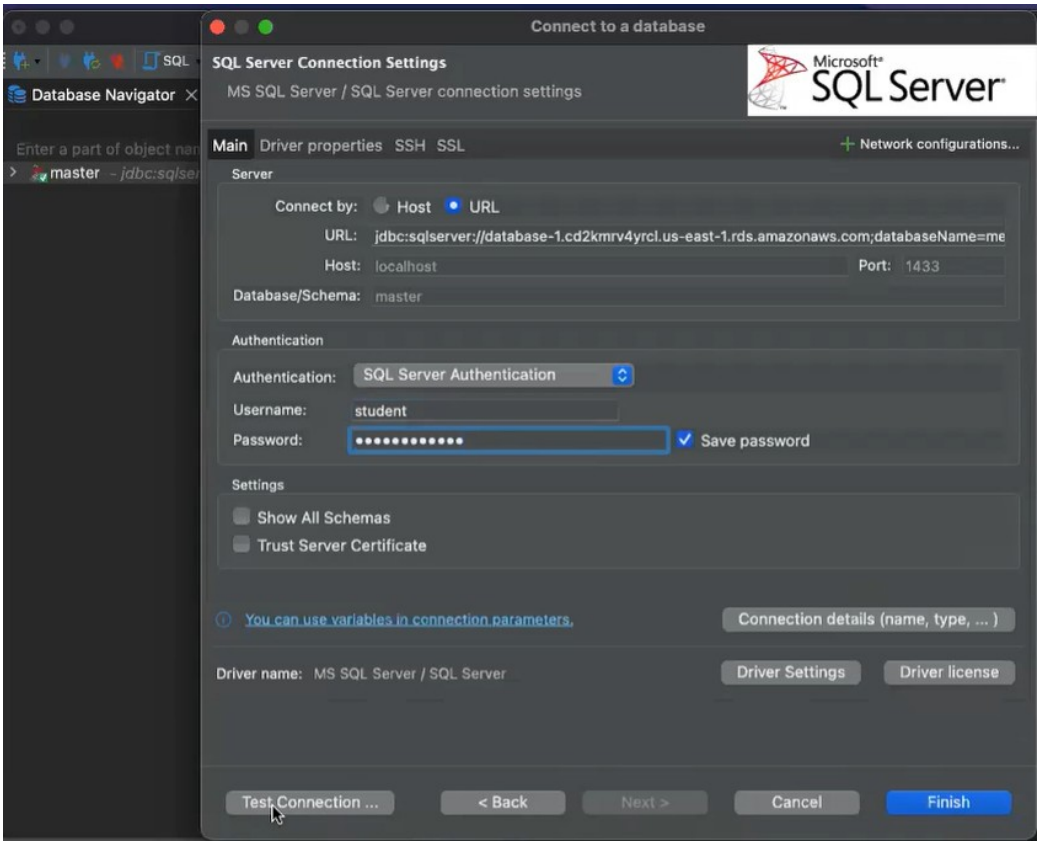
2.Click on next button.

3.Choose URL (It does not work with HOST), and paste the url, username and password from ui-config.properties.

4.Click onTest connection, then Finixh.

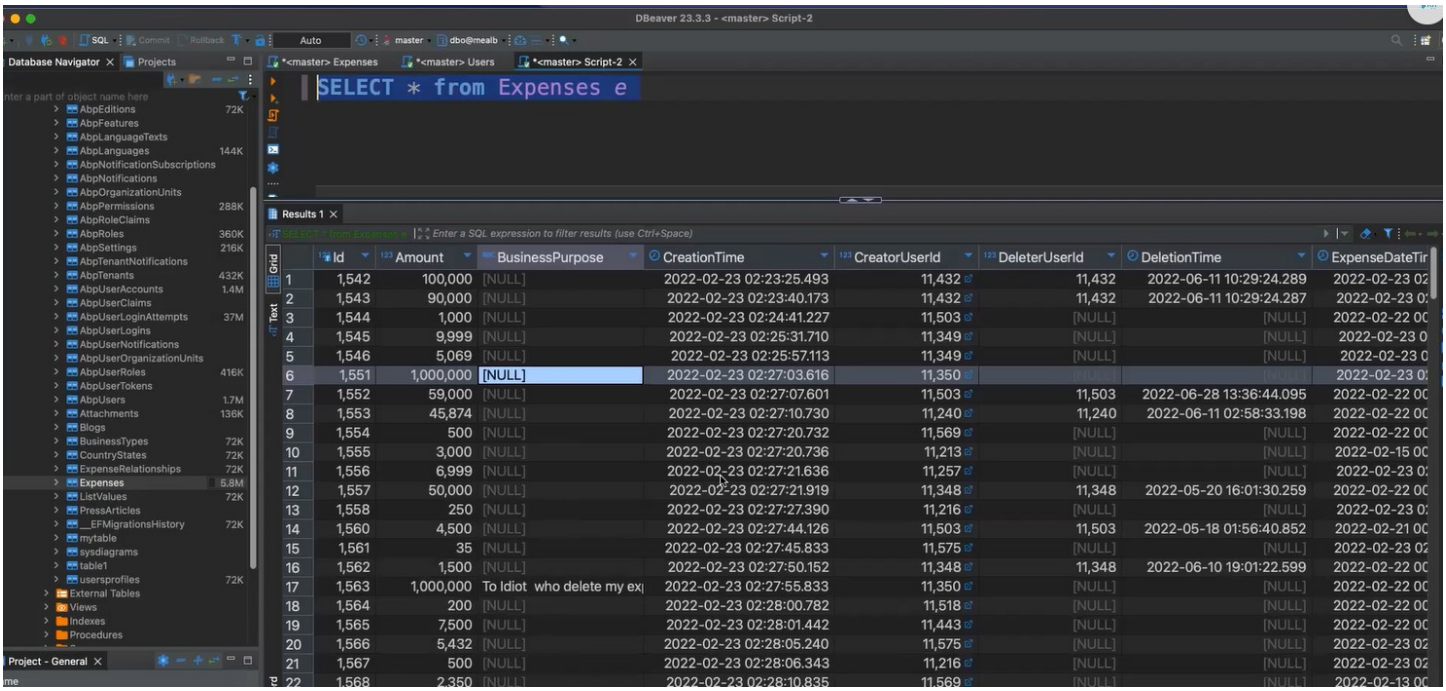
5. Go to dbo


SQL
Commit
Rollback
Database Navigator
Projects
Enter a part of object name here
master - jdbc:sqlserver://database-1.cd2kmrv4
Databases
mealb
Schemas
dbo
Tables
External Tables
Views
Indexes
Procedures
Sequences
Synonyms
Triggers
Data Types
Database triggers
rdsadmin
Security
Administer


Connect to a database
SQL Server Connection Settings
MS SQL Server / SQL Server connection settings
Microsoft SQL Server
Main
Driver properties
SSH
SSL
Network configurations...
Server
Connect by:
Host
URL
URL:
jdbc:sqlserver://database-1.cd2kmrv4yrcl.us-east-1.rds.amazonaws.com;databaseName=me
Host:
localhost
Port:
1433
Database/Schema:
master
Authentication
Authentication:
SQL Server Authentication
Username:
student
Password:
Save password
Settings
Show All Schemas
Trust Server Certificate
You can use variables in connection parameters.
Connection details (name, type, ...)
Driver name:
MS SQL Server / SQL Server
Driver Settings
Driver license
Test Connection ...
< Back
Next >
Cancel
Finish

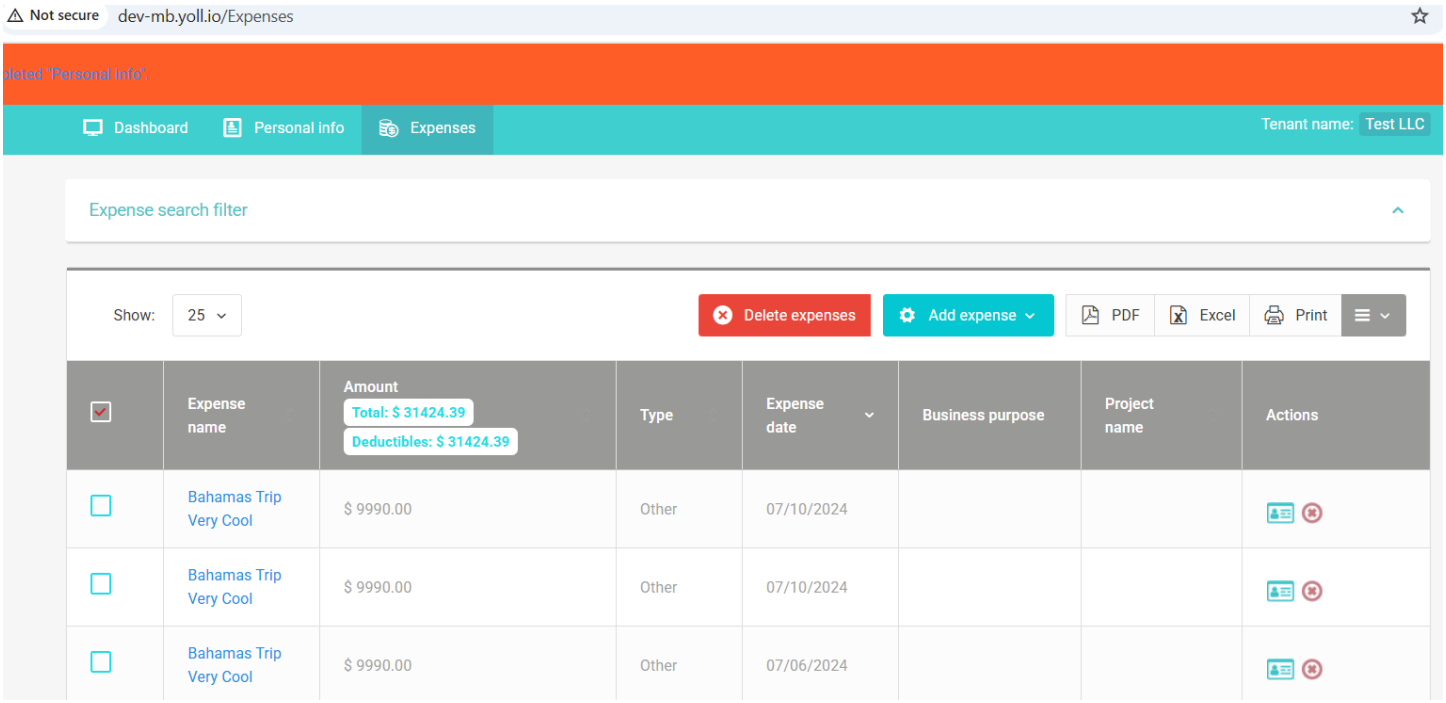
```
mysql.url: jdbc:mysql://dev-hrm.yoll.io:3306/orangehrm_mysql
yollhrm.database.username: root
yollhrm.database.password: Bootcamp5#
mb.database.url: jdbc:sqlserver://database-1.cd2kmrv4yrcl.us-east-1.rds.amazonaws.com;databaseName=mealb
mb.database.username: student
mb.database.password: Student2022Q
#HRM URL and valid credentials
yollhrm.url: http://dev-hrm.yoll.io/index.php/auth/login
yollhrm.username:yoll-student
yollhrm.password:Bootcamp5#

##File Uploadload URL
herokuapp.url:https://the-internet.herokuapp.com/

#Alert Page URL
#iFrames Page URL
iFrames.url: https://the-internet.herokuapp.com/iframe
```

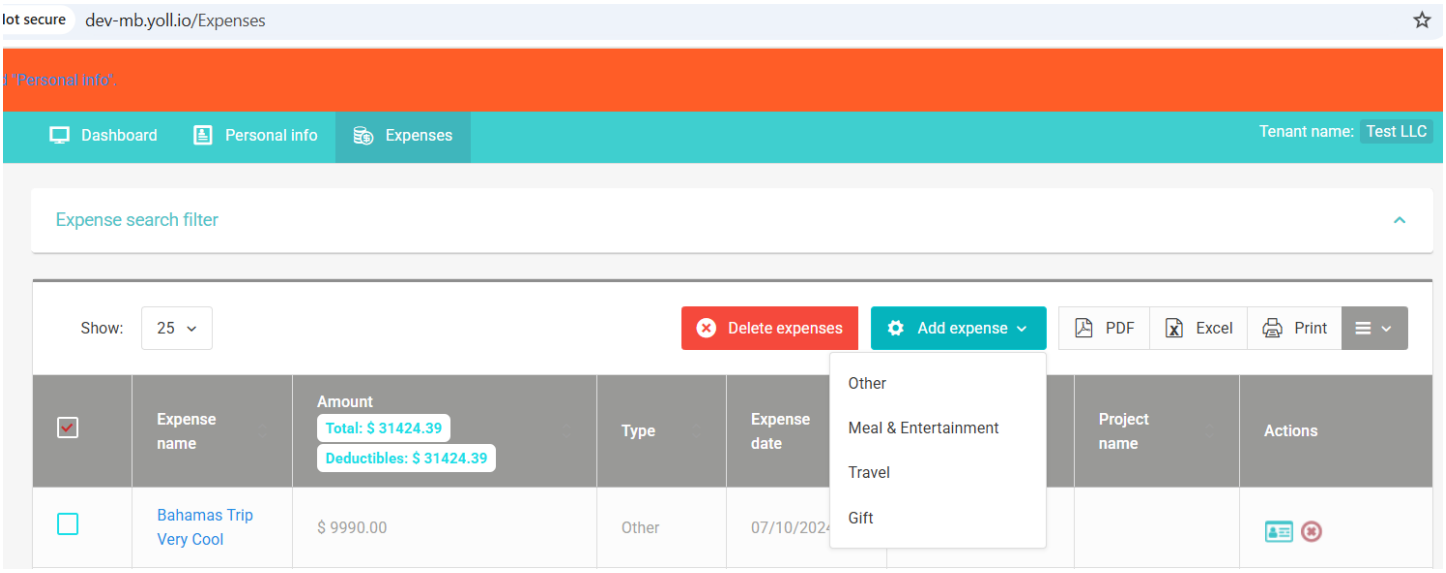
mb.database.url: jdbc:sqlserver://database-1.cd2kmrv4yrcl.us-east-1.rds.amazonaws.com;databaseName=mealb
mb.database.username: student
mb.database.password: Student2022Q

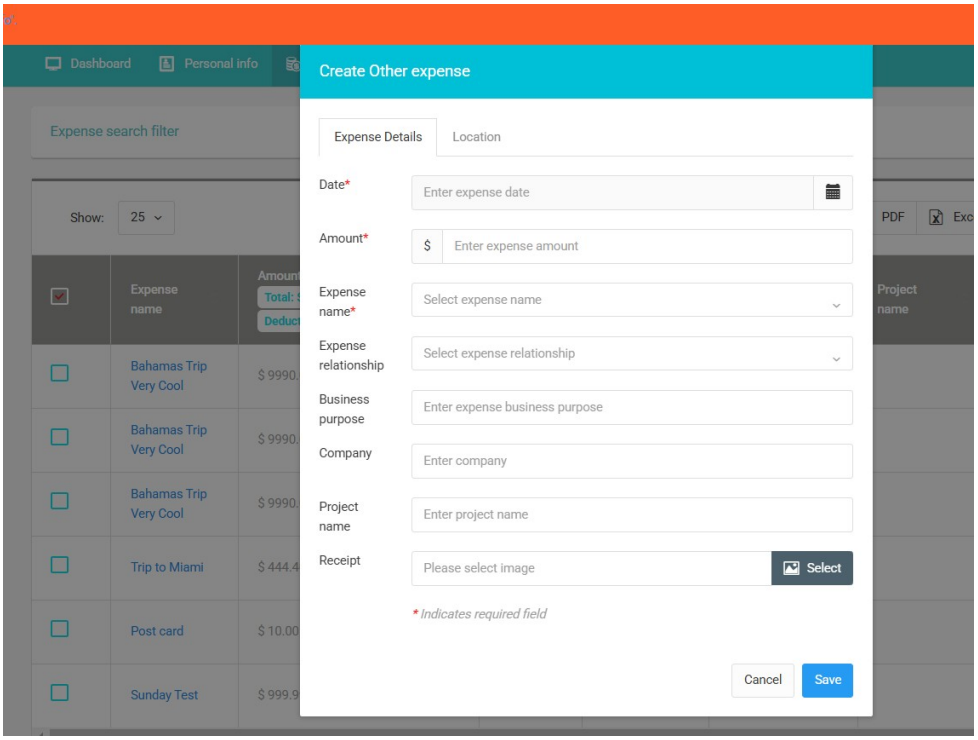


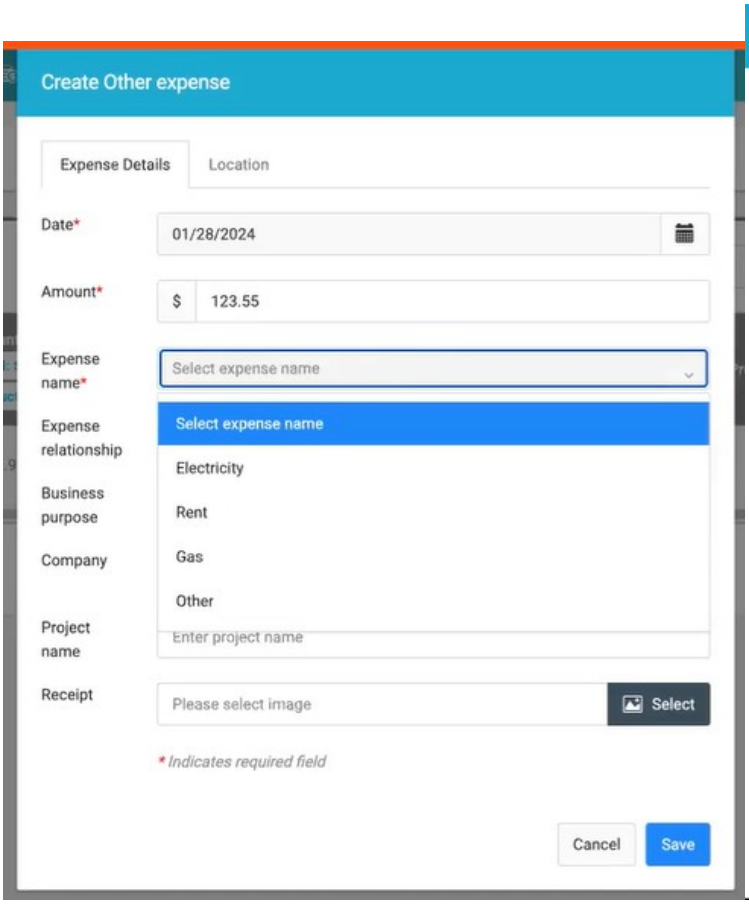

```
Background:
Given user navigates to MealB landing Page
And clicks on Sign In button
And logs in with employee account

Scenario: Add Expense
Given user is on the Dashboard page
When user clicks on the Expenses tab
And clicks on the Add Expense button
And selects "Other" as the expense type
```

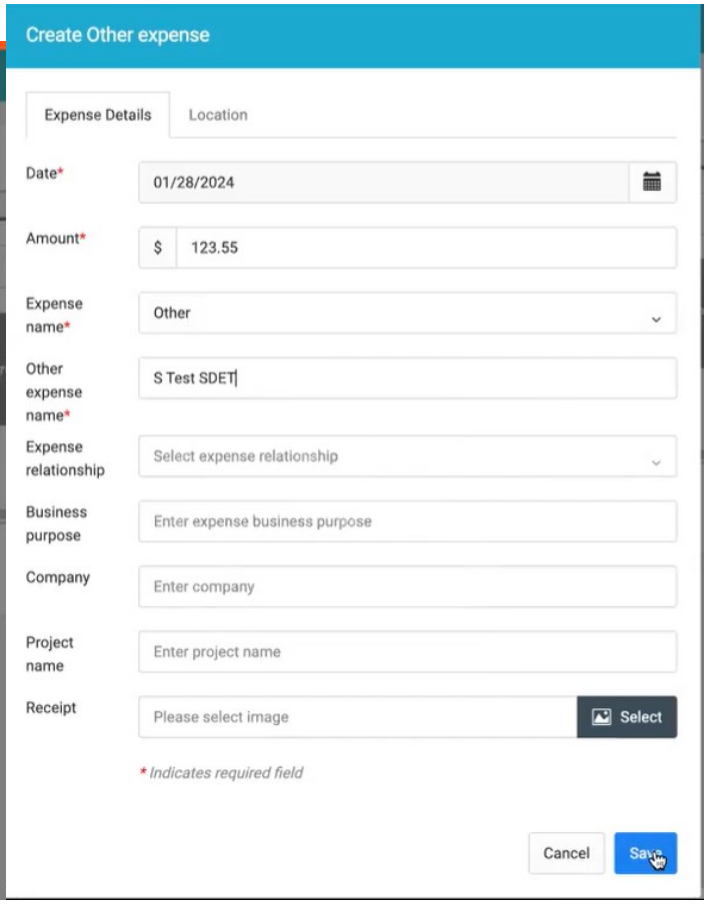
```
Then a modal pop-up is shown as a form
When user enters todays date as the expense date
And enters "123.55" as the expense amount
And selects "Other" as expense name
And enters "S Test SDET" as other expense name
And the user clicks the Save button
And the expense "S Test SDET" with amount "123.55" is listed on the Expenses page
And a record for "S Test SDET" with amount "123.55" is created in the database
```






Dashboard
Personal info
Expense search filter
Show: 25
PDF
Expense name
Project name
Bahamas Trip Very Cool
Bahamas Trip Very Cool
Bahamas Trip Very Cool
Trip to Miami
Post card
Sunday Test
Create Other expense
Expense Details
Location
Date*
Enter expense date
Amount*
$
Enter expense amount
Expense name*
Select expense name
Expense relationship
Select expense relationship
Business purpose
Enter expense business purpose
Company
Enter company
Project name
Enter project name
Receipt
Please select image
Select
* Indicates required field
Cancel
Save


Create Other expense
Expense Details
Location
Date*
01/28/2024
Amount*
$
123.55
Expense name*
Select expense name
Expense relationship
Business purpose
Company
Project name
Enter project name
Receipt
Please select image
Select
* Indicates required field
Cancel
Save
Select expense name
Electricity
Rent
Gas
Other

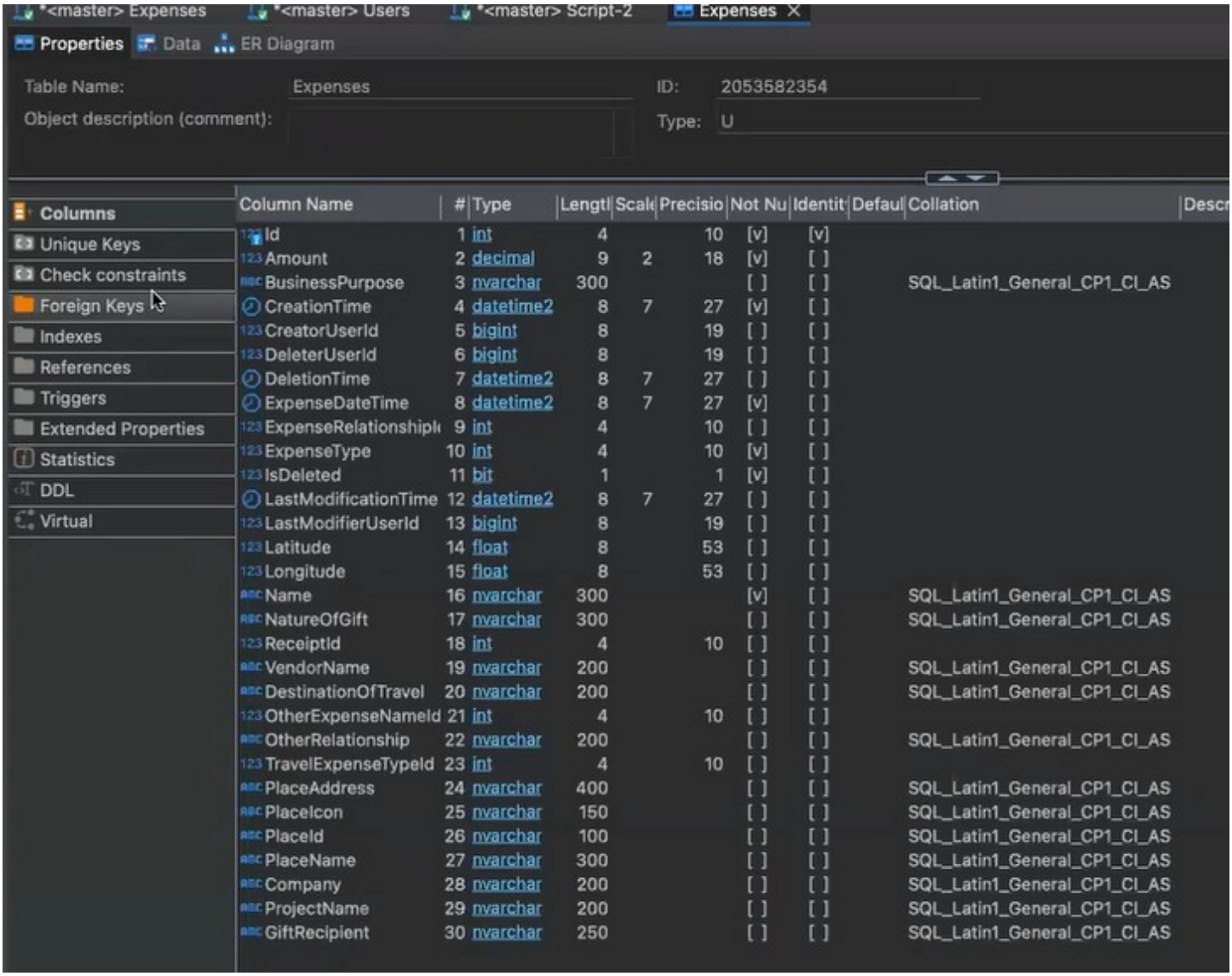

Create Other expense
Expense Details
Location
Date*
01/28/2024
Amount*
$
123.55
Expense name*
Other
Other expense name*
S Test SDET
Expense relationship
Select expense relationship
Business purpose
Enter expense business purpose
Company
Enter company
Project name
Enter project name
Receipt
Please select image
Select
* Indicates required field
Cancel
Save

Show: 50

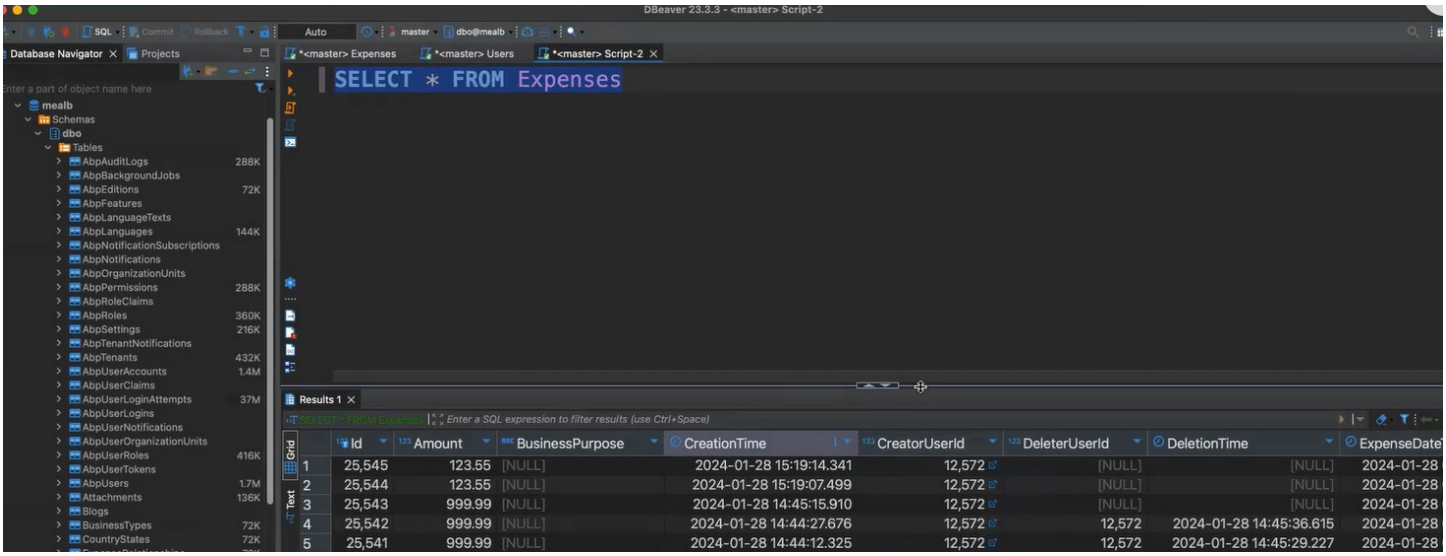
Add expense | PDF | Excel | Print

Expense name	Amount (Total: $ 1123.54, Deductibles: $ 1123.54)	Type	Expense date	Business purpose	Project name	Actions
Sunday Test	$ 999.99	Other	01/28/2024			
S Test SDET	$ 123.55	Other	01/28/2024			

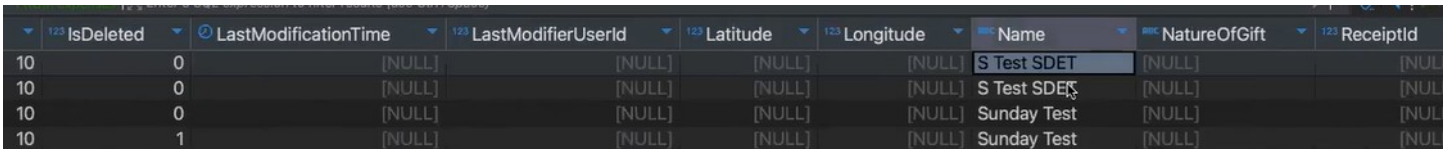
Columns name for Expense Table:



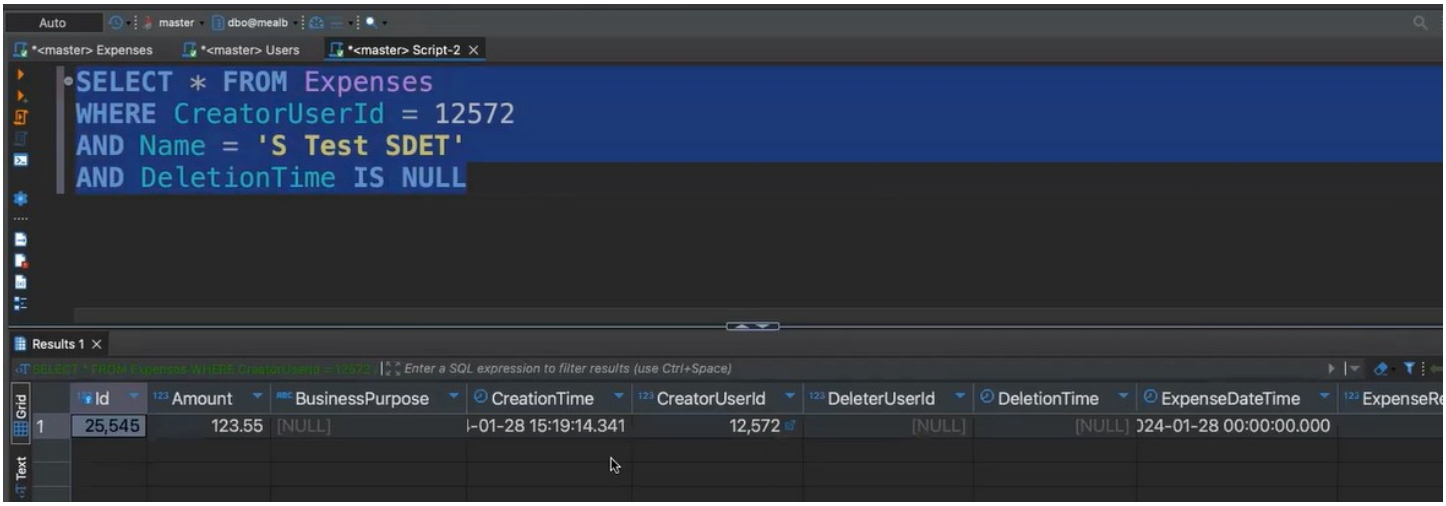
Column Name	#	Type	Length	Scale	Precision	Not Null	Identity	Default	Collation	Description
Id	1	int	4		10	[v]	[v]			
Amount	2	decimal	9	2	18	[v]	[ ]			
BusinessPurpose	3	nvarchar	300			[ ]	[ ]		SQL_Latin1_General_CP1_CI_AS	
CreationTime	4	datetime2	8	7	27	[v]	[ ]			
CreatorUserId	5	bigint	8		19	[ ]	[ ]			
DeleterUserId	6	bigint	8		19	[ ]	[ ]			
DeletionTime	7	datetime2	8	7	27	[ ]	[ ]			
ExpenseDateTime	8	datetime2	8	7	27	[v]	[ ]			
ExpenseRelationshipI	9	int	4		10	[ ]	[ ]			
ExpenseType	10	int	4		10	[v]	[ ]			
IsDeleted	11	bit	1		1	[v]	[ ]			
LastModificationTime	12	datetime2	8	7	27	[ ]	[ ]			
LastModifierUserId	13	bigint	8		19	[ ]	[ ]			
Latitude	14	float	8		53	[ ]	[ ]			
Longitude	15	float	8		53	[ ]	[ ]			
Name	16	nvarchar	300			[v]	[ ]		SQL_Latin1_General_CP1_CI_AS	
NatureOfGift	17	nvarchar	300			[ ]	[ ]		SQL_Latin1_General_CP1_CI_AS	
ReceiptId	18	int	4		10	[ ]	[ ]			
VendorName	19	nvarchar	200			[ ]	[ ]		SQL_Latin1_General_CP1_CI_AS	
DestinationOfTravel	20	nvarchar	200			[ ]	[ ]		SQL_Latin1_General_CP1_CI_AS	
OtherExpenseNameId	21	int	4		10	[ ]	[ ]			
OtherRelationship	22	nvarchar	200			[ ]	[ ]		SQL_Latin1_General_CP1_CI_AS	
TravelExpenseTypeId	23	int	4		10	[ ]	[ ]			
PlaceAddress	24	nvarchar	400			[ ]	[ ]		SQL_Latin1_General_CP1_CI_AS	
PlaceIcon	25	nvarchar	150			[ ]	[ ]		SQL_Latin1_General_CP1_CI_AS	
PlaceId	26	nvarchar	100			[ ]	[ ]		SQL_Latin1_General_CP1_CI_AS	
PlaceName	27	nvarchar	300			[ ]	[ ]		SQL_Latin1_General_CP1_CI_AS	
Company	28	nvarchar	200			[ ]	[ ]		SQL_Latin1_General_CP1_CI_AS	
ProjectName	29	nvarchar	200			[ ]	[ ]		SQL_Latin1_General_CP1_CI_AS	
GiftRecipient	30	nvarchar	250			[ ]	[ ]		SQL_Latin1_General_CP1_CI_AS	



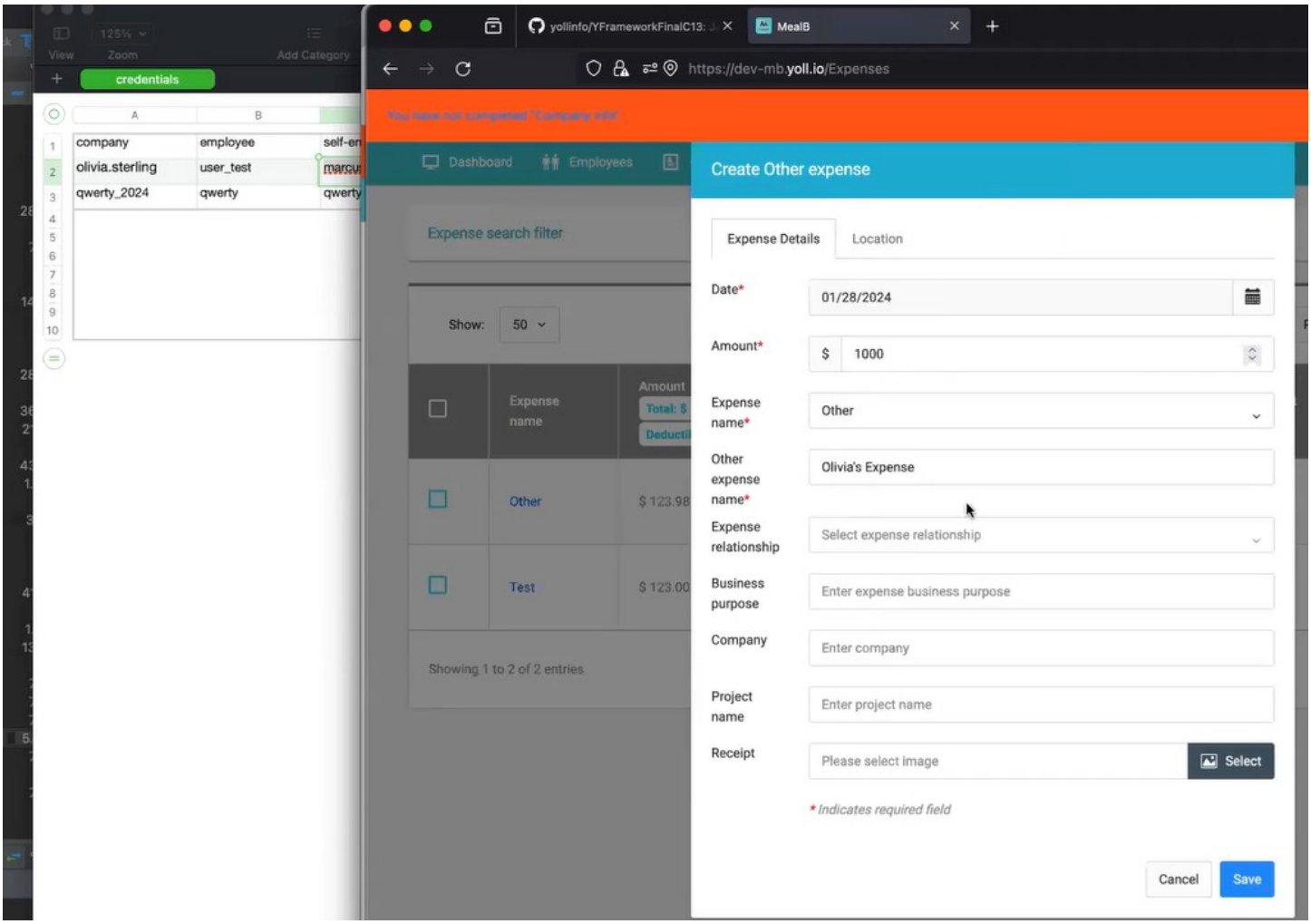
Name:


IsDeleted LastModificationTime LastModifierUserId Latitude Longitude Name NatureOfGift ReceiptId
10 0 [NULL] [NULL] [NULL] [NULL] S Test SDET [NULL]
10 0 [NULL] [NULL] [NULL] [NULL] S Test SDET [NULL]
10 0 [NULL] [NULL] [NULL] [NULL] Sunday Test [NULL]
10 1 [NULL] [NULL] [NULL] [NULL] Sunday Test [NULL]

Using filtration: Name=’S Test SDET’ and it is already deleted. The cell used:DeletionTime is NULL

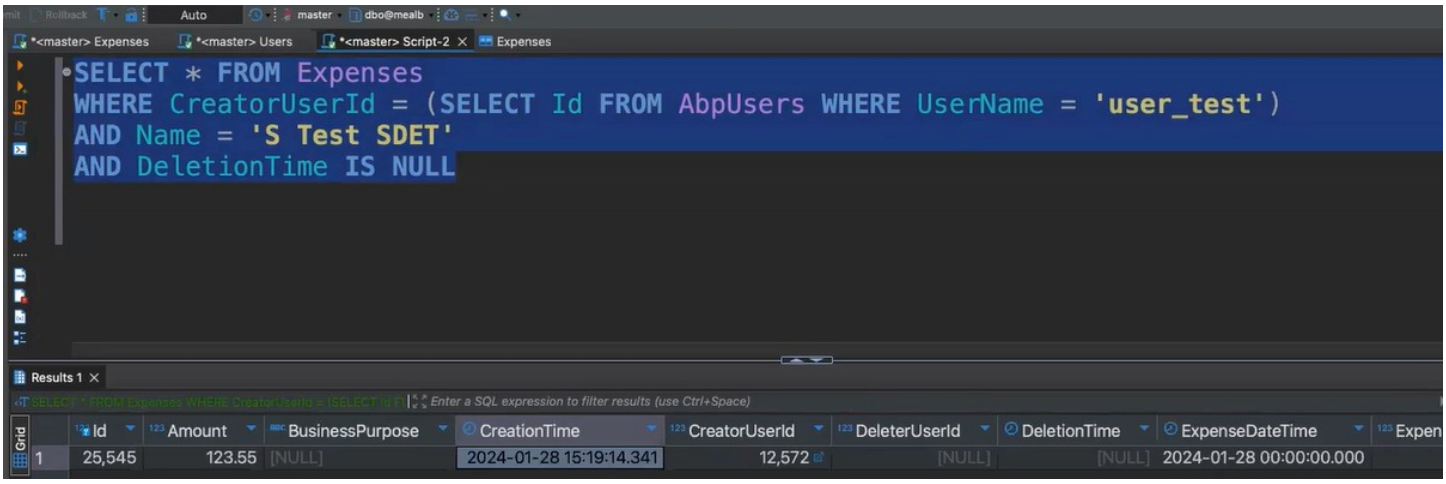

SELECT * FROM Expenses
WHERE CreatorUserId = 12572
AND Name = 'S Test SDET'
AND DeletionTime IS NULL
Results 1
Id Amount BusinessPurpose CreationTime CreatorUserId DeleterUserId DeletionTime ExpenseDateTime
1 25,545 123.55 [NULL] -01-28 15:19:14.341 12,572 [NULL] [NULL] 024-01-28 00:00:00.000

Creating an expense for Olivia.Sterling:

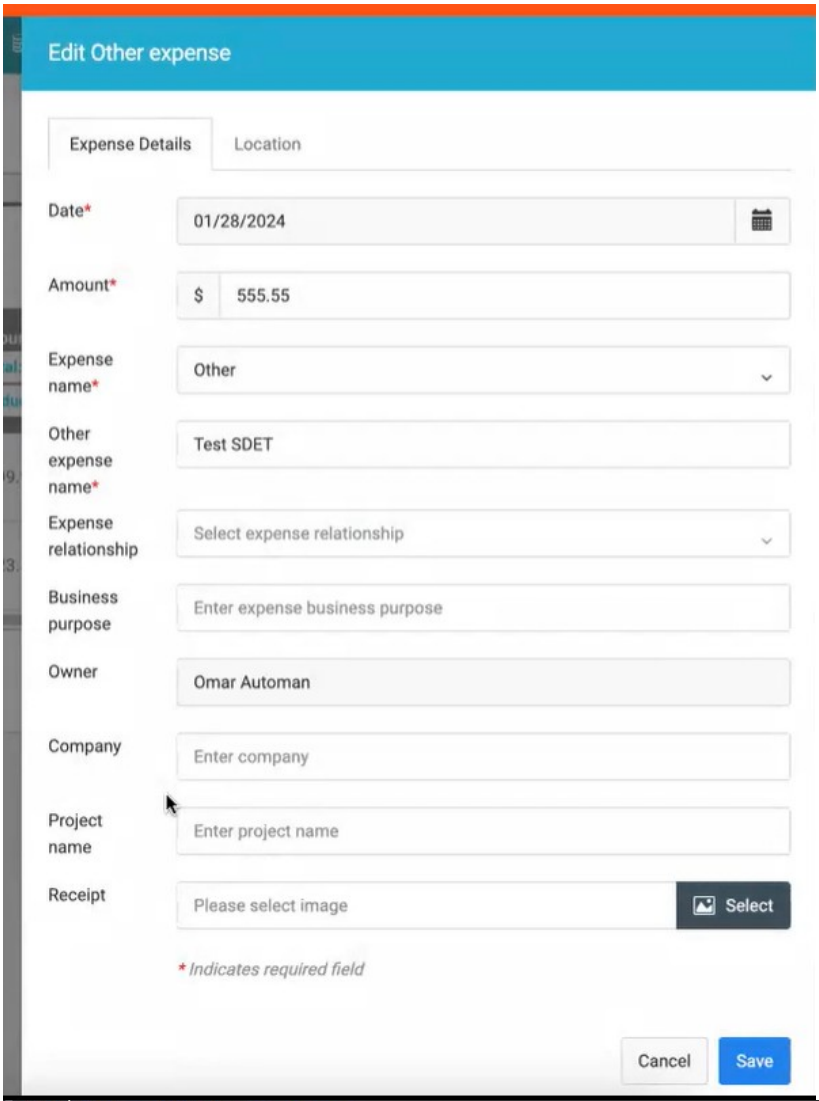


Expense name	Amount Total: $ 1246.98 Deductibles: $ 1246.98	Type	Expense date	Business purpose	Project name	Owner	Actions
Olivia's Expense	$ 1000.00	Other	01/28/2024			Olivia Dynamics LLC	
Other	$ 123.98	Other	01/13/2024			Olivia Dynamics LLC	
Test	$ 123.00	Meal & Entertainment	01/10/2024			Olivia Dynamics LLC	

```sql
SELECT * FROM Expenses
WHERE CreatorUserId = (SELECT Id FROM AbpUsers WHERE UserName = 'olivia.sterling')
AND Name = 'Olivias Expense'
AND DeletionTime IS NULL
```




```
Scenario: Update Expense
Given user is on the Dashboard page
When user clicks on the Expenses tab
And clicks on the Add Expense button
And user selects an existing expense "S Test SDET" with amount "123.55"
And user clicks on the Edit button
Then a modal pop-up is shown with the expense details
When user updates the amount to "555.55"
And user updates the expense name to "Test SDET"
And the user clicks the Save button

Then the expense "Test SDET" with amount "555.55" is updated on the Expenses page
And a record for "Test SDET" with amount "555.55" is updated in the database
```

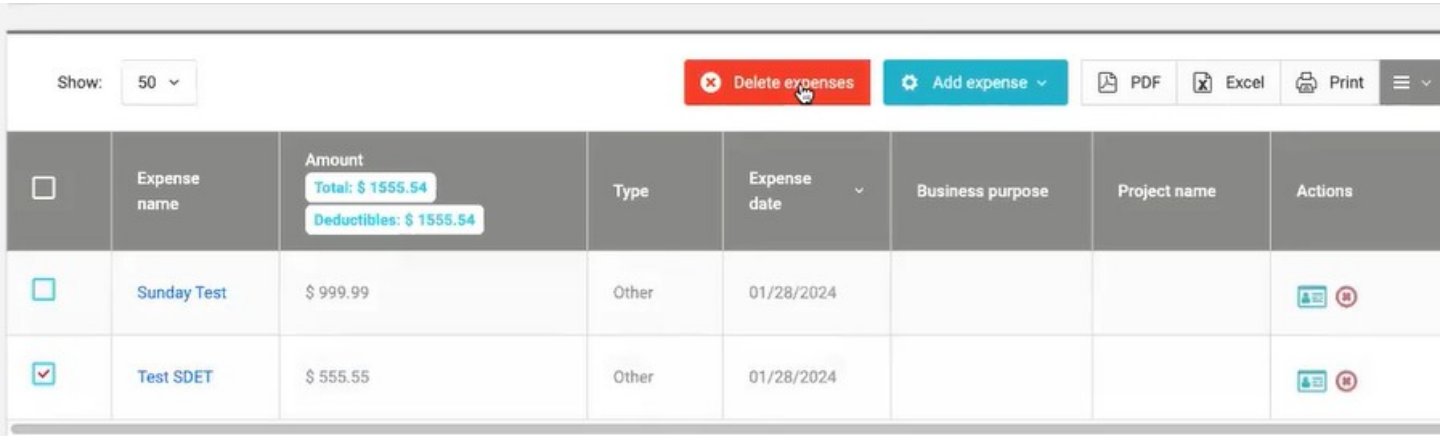


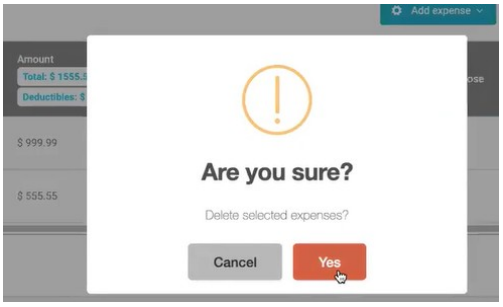

```
SELECT * FROM Expenses
WHERE CreatorUserId = (SELECT Id FROM AbpUsers WHERE UserName = 'user_test')
AND Name = 'Test SDET'
AND Amount = 555.55
AND DeletionTime IS NULL
AND LastModificationTime IS NOT NULL
```

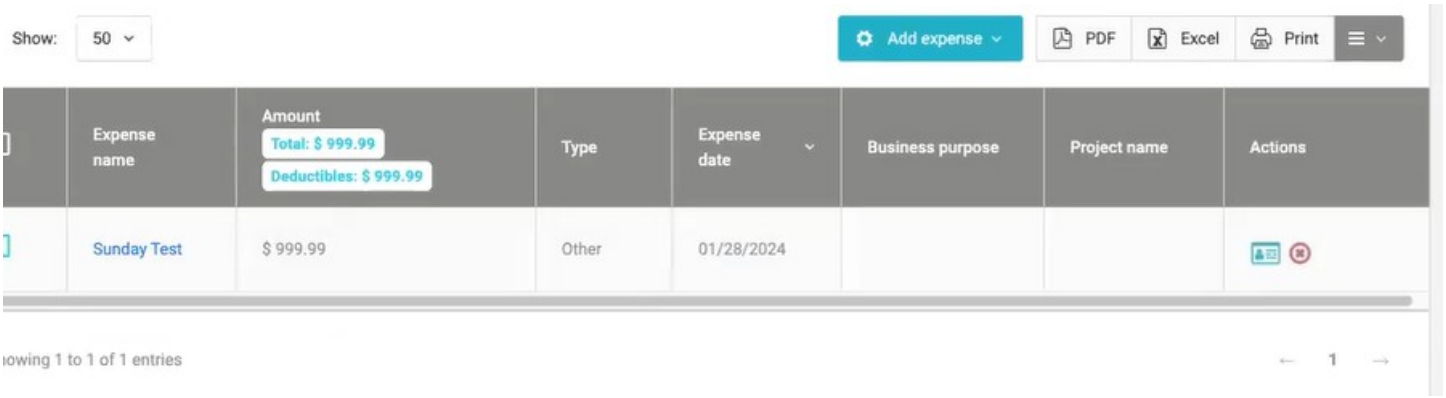
Enter a SQL expression to filter results (use Ctrl+Space)

DeleterUserId	DeletionTime	ExpenseDateTime	ExpenseRelationshipId	ExpenseType	IsDeleted	LastModificationTime	LastModifierUserId
[NULL]	[NULL]	2024-01-28 00:00:00.000	[NULL]	10	0	2024-01-28 15:41:54.640	12,572

Deleting Test SDET







To verify the delete, the deletion should not be null

The record shows the time of deletion: DeletionTime

```
SELECT * FROM Expenses
WHERE CreatorUserId = (SELECT Id FROM AbpUsers WHERE UserName = 'user_test')
AND Name = 'Test SDET'
AND Amount = 555.55
AND DeletionTime IS NOT NULL
```

Id	Amount	BusinessPurpose	CreationTime	CreatorUserId	DeleterUserId	DeletionTime	ExpenseDateTime	ExpenseRelatio
25,545	555.55	[NULL]	2024-01-28 15:19:14.341	12,572	12,572	01-28 15:46:25.239	2024-01-28 00:00:00.000	

Make sure to have on pom.xml: Microsoft SQL

```
<dependency>
    <groupId>com.microsoft.sqlserver</groupId>
    <artifactId>mssql-jdbc</artifactId>
    <version>8.2.2.jre8</version>
</dependency>
```

```
<groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.18</version>
</dependency>
```

Execution:

```
package ui_automation.utilities;

import constants.Constants;

public class PracticeDBConnection {

    public static void main(String[] args) {

        // To create a connection to the database and execute the SQL Queries we need to:

        // 1. Load and register the driver for the specific DB to you JVM
        // 2. Create and establish the DB Connection
        // 3. Create a SQL Statement
        // 4. Execute the statement and get the results
        // 5. Process the results (For Queries)
        // 6. Close all the connections
        // Get the url, username and password in your code

        String url = ConfigurationReader.getProperty(Constants.UI_CONFIG,
Constants.MEALB_DB_URL);
        String username = ConfigurationReader.getProperty(Constants.UI_CONFIG,
Constants.MEALB_DB_USERNAME);
        String password = ConfigurationReader.getProperty(Constants.UI_CONFIG,
Constants.MEALB_DB_PASSWORD);

        System.out.println(url);
        System.out.println(username);
        System.out.println(password);
    }
}
```

```
jdbc:sqlserver://database-1.cd2kmrv4yrcl.us-east-1.rds.amazonaws.com;databaseName=mealb
student
Student2022Q

Process finished with exit code 0
```

// Print CreatorUserId=12572

```
if(creatorId==12572){
    System.out.println("| ID | "+ id+"| Name | "+name+"| Amount | "+amount+"| Creationtime |"+creationtime+"| CreatorUserId | "+creatorId);

}
```

```
jdbc:sqlserver://database-1.cd2kmrv4yrcl.us-east-1.rds.amazonaws.com;databaseName=mealb
student
Student2022Q
com.microsoft.sqlserver.jdbc.SQLServerDriver
| ID | 24297| Name | Travel to NY| Amount | 1250.0| Creationtime |2023-10-21 04:12:45.0826902| CreatorUserId | 12572
======================
| ID | 24298| Name | Business Lunch| Amount | 200.0| Creationtime |2023-10-21 04:13:20.6906890| CreatorUserId | 12572
======================
| ID | 24299| Name | Customer Christmas Gift| Amount | 5000.0| Creationtime |2023-10-21 04:14:07.8172810| CreatorUserId | 12572
======================
| ID | 24300| Name | Electricity| Amount | 300.0| Creationtime |2023-10-21 04:14:33.6374253| CreatorUserId | 12572
======================
| ID | 24301| Name | Rent| Amount | 600.0| Creationtime |2023-10-21 04:15:24.2744516| CreatorUserId | 12572
======================
| ID | 24833| Name | Flight ticket| Amount | 125.0| Creationtime |2023-12-13 02:17:03.6602046| CreatorUserId | 12572
======================
| ID | 24845| Name | Christmas Gift| Amount | 55.0| Creationtime |2023-12-14 21:55:42.9426355| CreatorUserId | 12572
```

```
package ui_automation.utilities;

import constants.Constants;

import java.sql.*;

public class PracticeDBConnection {

    public static void main(String[] args) {

        // To create a connection to the database and execute the SQL Queries we need to:

        // 1. Load and register the driver for the specific DB to you JVM
        // 2. Create and establish the DB Connection
        // 3. Create a SQL Statement
        // 4. Execute the statement and get the results
        // 5. Process the results (For Queries)
        // 6. Close all the connections
        // Get the url, username and password in your code
        String url = ConfigurationReader.getProperty(Constants.UI_CONFIG,
Constants.MEALB_DB_URL);
        String username = ConfigurationReader.getProperty(Constants.UI_CONFIG,
Constants.MEALB_DB_USERNAME);
```

```
        String password = ConfigurationReader.getProperty(Constants.UI_CONFIG,
Constants.MEALB_DB_PASSWORD);
        String dbDriver = ConfigurationReader.getProperty(Constants.UI_CONFIG,
Constants.MEALB_DB_DRIVER);

        System.out.println(url);
        System.out.println(username);
        System.out.println(password);
        System.out.println(dbDriver);
        // 1. Load and register the driver for the specific DB to you JVM
        try {
            Class.forName(dbDriver);
        } catch (ClassNotFoundException e) {
            throw new RuntimeException(e);
        }

        try {
            // 2. Create and establish the DB Connection
            //connection must be sql java
            Connection connection= DriverManager.getConnection(url,username,password);

            // 3. Create a SQL Statement
            Statement statement=connection.createStatement();
            // 4. Execute the statement and get the results
              //Create the query first
              //Execute it
            String query="SELECT *FROM Expenses";
            statement.executeQuery(query);
            ResultSet resultSet=statement.executeQuery(query);

            // 5. Process the results (For Queries)
            while(resultSet.next()){
                int id =resultSet.getInt("ID");
                double amount =resultSet.getDouble("Amount");
                String name =resultSet.getString("Name");
                int creatorId=resultSet.getInt("CreatorUserId");
                String creationtime=resultSet.getString("Creationtime");

                   // Print CreatorUserId=12572
                if(creatorId==12572){
System.out.println("| ID | "+ id+"| Name | "+name+"| Amount | "+amount+"|
Creationtime |"+creationtime+"| CreatorUserId | "+creatorId);
                    System.out.println("======================");
                }


                   // Print All records
System.out.println("| ID | "+ id+"| Name | "+name+"| Amount | "+amount+"|
Creationtime |"+creationtime+"| CreatorUserId | "+creatorId);


            }

        } catch(SQLException e){
        e.printStackTrace();

    }
}}
                   // Print CreatorUserId=12572 and name = Sunday Test
```

```
            if(creatorId==12572 && name.equals("Sunday Test")){
                    System.out.println("| ID | "+ id+"| Name | "+name+"| 
Amount | "+amount+"| Creationtime |"+creationtime+"| CreatorUserId | 
"+creatorId);
                    System.out.println("======================");
                }
```

```
| ID | 25541| Name | Sunday Test| Amount | 999.99| Creationtime |2024-01-28 14:44:12.3258411| CreatorUserId | 12572
======================
| ID | 25542| Name | Sunday Test| Amount | 999.99| Creationtime |2024-01-28 14:44:27.6769827| CreatorUserId | 12572
======================
| ID | 25543| Name | Sunday Test| Amount | 999.99| Creationtime |2024-01-28 14:45:15.9107680| CreatorUserId | 12572
======================
| ID | 27664| Name | Sunday Test| Amount | 999.99| Creationtime |2024-07-04 02:00:12.8035903| CreatorUserId | 12572
======================
```

Updated Approach :

```
package ui_automation.utilities;

import constants.Constants;

import java.sql.*;

public class PracticeDBConnection {

    public static void main(String[] args) {

        // To create a connection to the database and execute the SQL Queries we need to:

        // 1. Load and register the driver for the specific DB to you JVM
        // 2. Create and establish the DB Connection
        // 3. Create a SQL Statement
        // 4. Execute the statement and get the results
        // 5. Process the results (For Queries)
        // 6. Close all the connections
        // Get the url, username and password in your code
        String url = ConfigurationReader.getProperty(Constants.UI_CONFIG,
Constants.MEALB_DB_URL);
        String username = ConfigurationReader.getProperty(Constants.UI_CONFIG,
Constants.MEALB_DB_USERNAME);
        String password = ConfigurationReader.getProperty(Constants.UI_CONFIG,
Constants.MEALB_DB_PASSWORD);
        String dbDriver = ConfigurationReader.getProperty(Constants.UI_CONFIG,
Constants.MEALB_DB_DRIVER);

        System.out.println(url);
        System.out.println(username);
        System.out.println(password);
        System.out.println(dbDriver);
        // 1. Load and register the driver for the specific DB to you JVM
        try {
            Class.forName(dbDriver);
        } catch (ClassNotFoundException e) {
            throw new RuntimeException(e);
```

```
        }
        //Create the query first
        String query = "SELECT *FROM Expenses";
        try (
                // 2. Create and establish the DB Connection
                //connection must be sql java
                Connection connection = DriverManager.getConnection(url, username,
password);
                // 3. Create a SQL Statement
                Statement statement = connection.createStatement();
                // 4. Execute the statement and get the results
                //Create the query first
                //Execute it
                ResultSet resultSet = statement.executeQuery(query);
        ) {


            // 5. Process the results (For Queries)
            while (resultSet.next()) {
                int id = resultSet.getInt("ID");
                double amount = resultSet.getDouble("Amount");
                String name = resultSet.getString("Name");
                int creatorId = resultSet.getInt("CreatorUserId");
                String creationtime = resultSet.getString("Creationtime");

                // Print CreatorUserId=12572 and name = Sunday Test
                if (creatorId == 12572 && name.equals("Sunday Test")) {
                    System.out.println("| ID | " + id + "| Name | " + name + "| Amount |
" + amount + "| Creationtime |" + creationtime + "| CreatorUserId | " + creatorId);
                    System.out.println("======================");
                }

                // Print All records
                // System.out.println("| ID | "+ id+"| Name | "+name+"| Amount |
"+amount+"| Creationtime |"+creationtime+"| CreatorUserId | "+creatorId);

            }

        } catch (SQLException e) {
            e.printStackTrace();

        }
    }
}
```

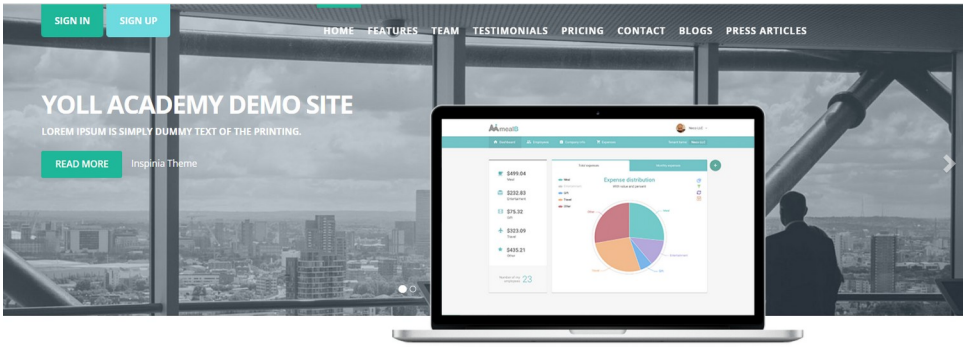
Task

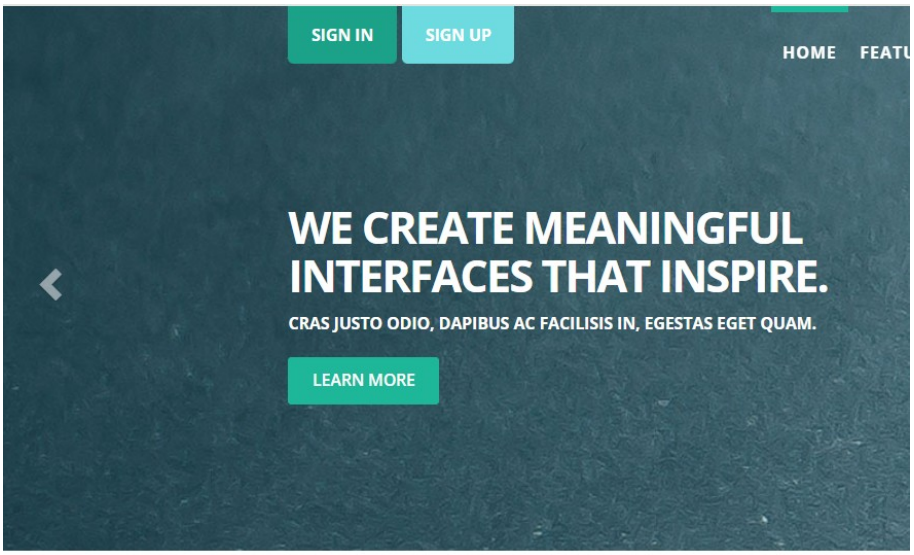
```
@dbtest
Feature: MealB Expenses
  Background:
    Given user navigates to MealB landing Page
    And clicks on Sign In button
    And logs in with employee account
  Scenario: Add Expense
    Given user is on the Dashboard page
    When user clicks on the Expenses tab
    And clicks on the Add Expense button
    And selects "Other" as the expense type
    Then a modal pop-up is shown as a form
```

```
When user enters todays date as the expense date
And enters 123.55" as the expense amount
And selects "Other" as expense name
And enters "S Test SDET" as other expense name
And the user clicks the Save button
And the expense "S Test SDET" with amount "123.55" is listed on the Expenses page
And a record for "s Test SDET SDET" with amount "123.55" is created in the database
```

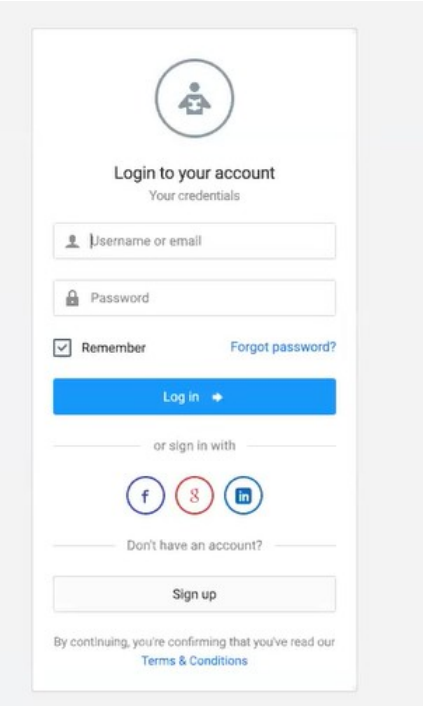
We need many pages besides the MealBExpenseCommonSteps class:

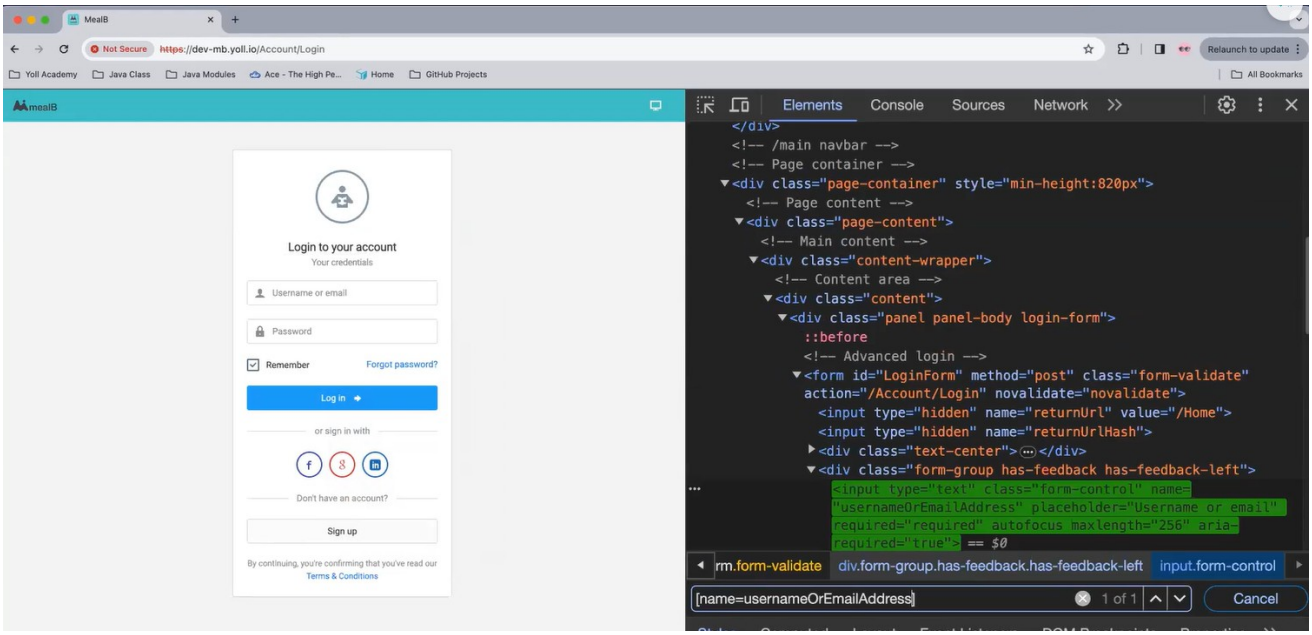
The main page: MealBLandingPage





The Loging Page: MealBLogingPage


Login to your account
Your credentials
Username or email
Password
Remember
Forgot password?
Log in
or sign in with
Don't have an account?
Sign up
By continuing, you're confirming that you've read our
Terms & Conditions


MealB
Not Secure
https://dev-mb.yoll.io/Account/Login
Relaunch to update
Yoll Academy
Java Class
Java Modules
Ace - The High Pe...
Home
GitHub Projects
All Bookmarks
Elements
Console
Sources
Network
Login to your account
Your credentials
Username or email
Password
Remember
Forgot password?
Log in
or sign in with
Don't have an account?
Sign up
By continuing, you're confirming that you've read our
Terms & Conditions

<!-- /main navbar -->
<!-- Page container -->

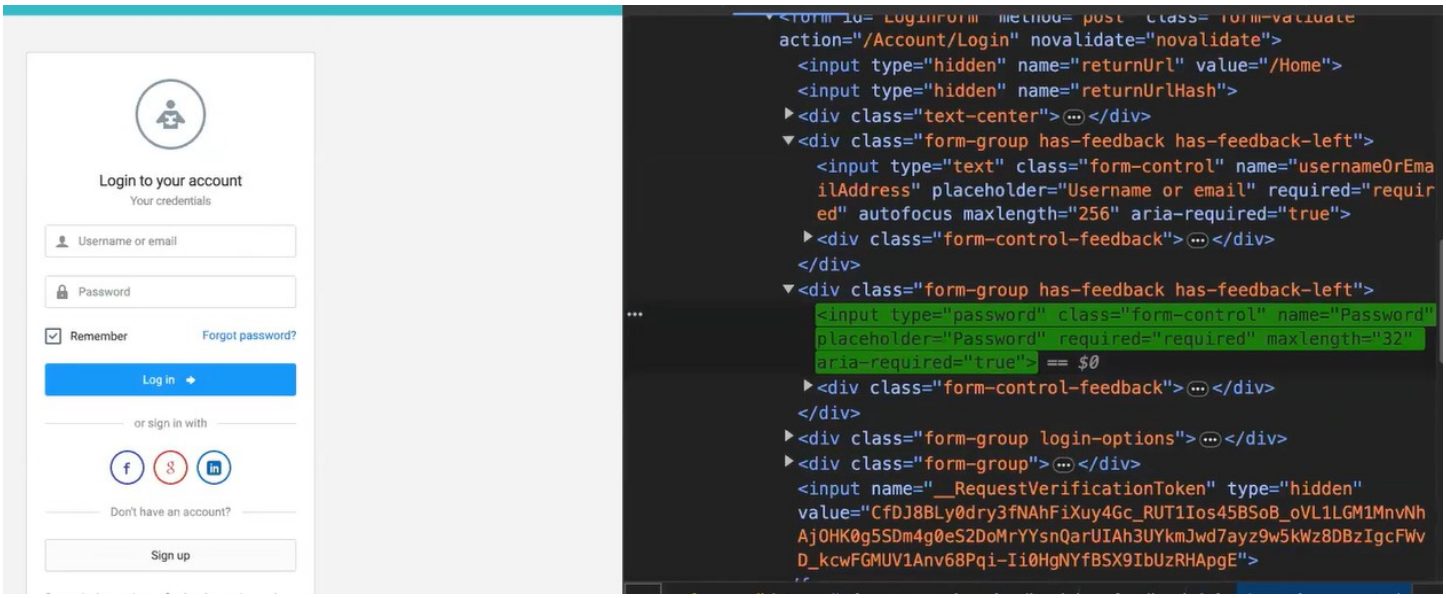
<!-- Page content -->

<!-- Main content -->

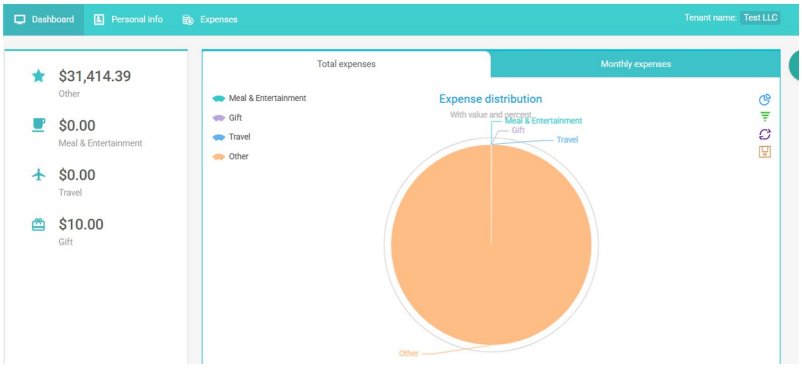
<!-- Content area -->

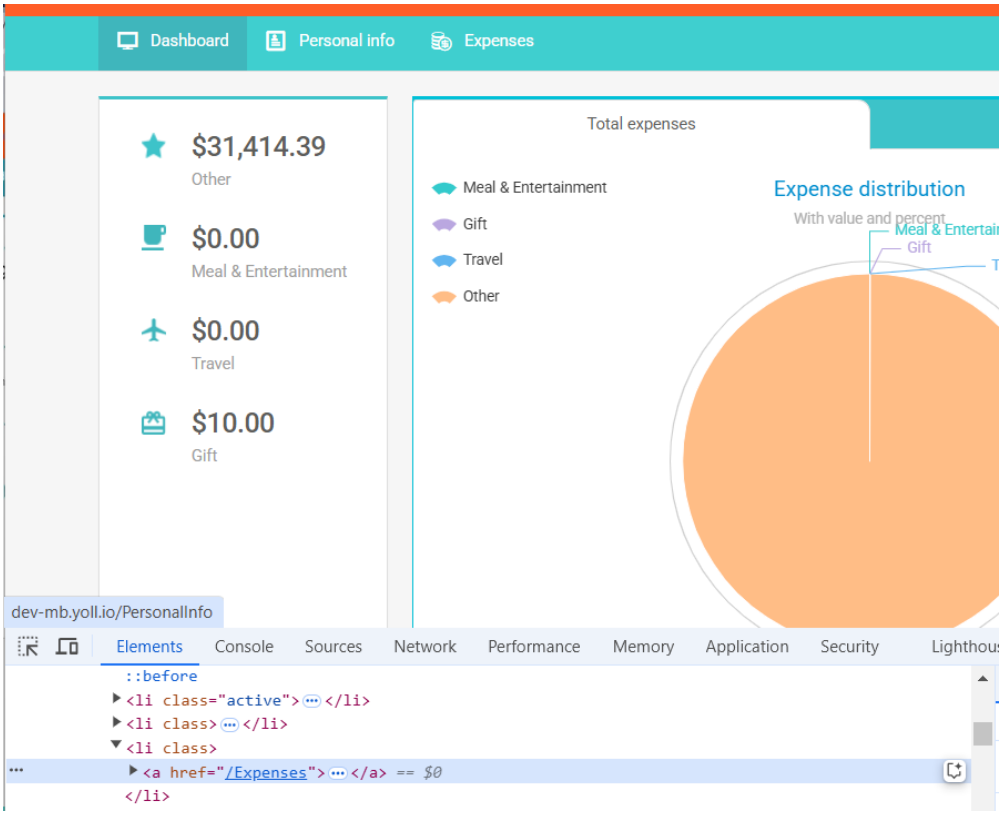
::before
<!-- Advanced login -->
<form id="LoginForm" method="post" class="form-validate"
action="/Account/Login" novalidate="novalidate">
<input type="hidden" name="returnUrl" value="/Home">
<input type="hidden" name="returnUrlHash">
…

<input type="text" class="form-control" name=
"usernameOrEmailAddress" placeholder="Username or email"
required="required" autofocus maxlength="256" aria-
required="true"> == $0
rm.form-validate
div.form-group.has-feedback.has-feedback-left
input.form-control
[name=usernameOrEmailAddress]
1 of 1
Cancel

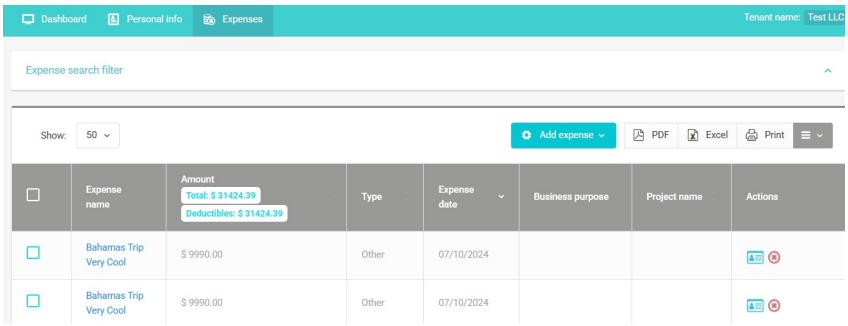


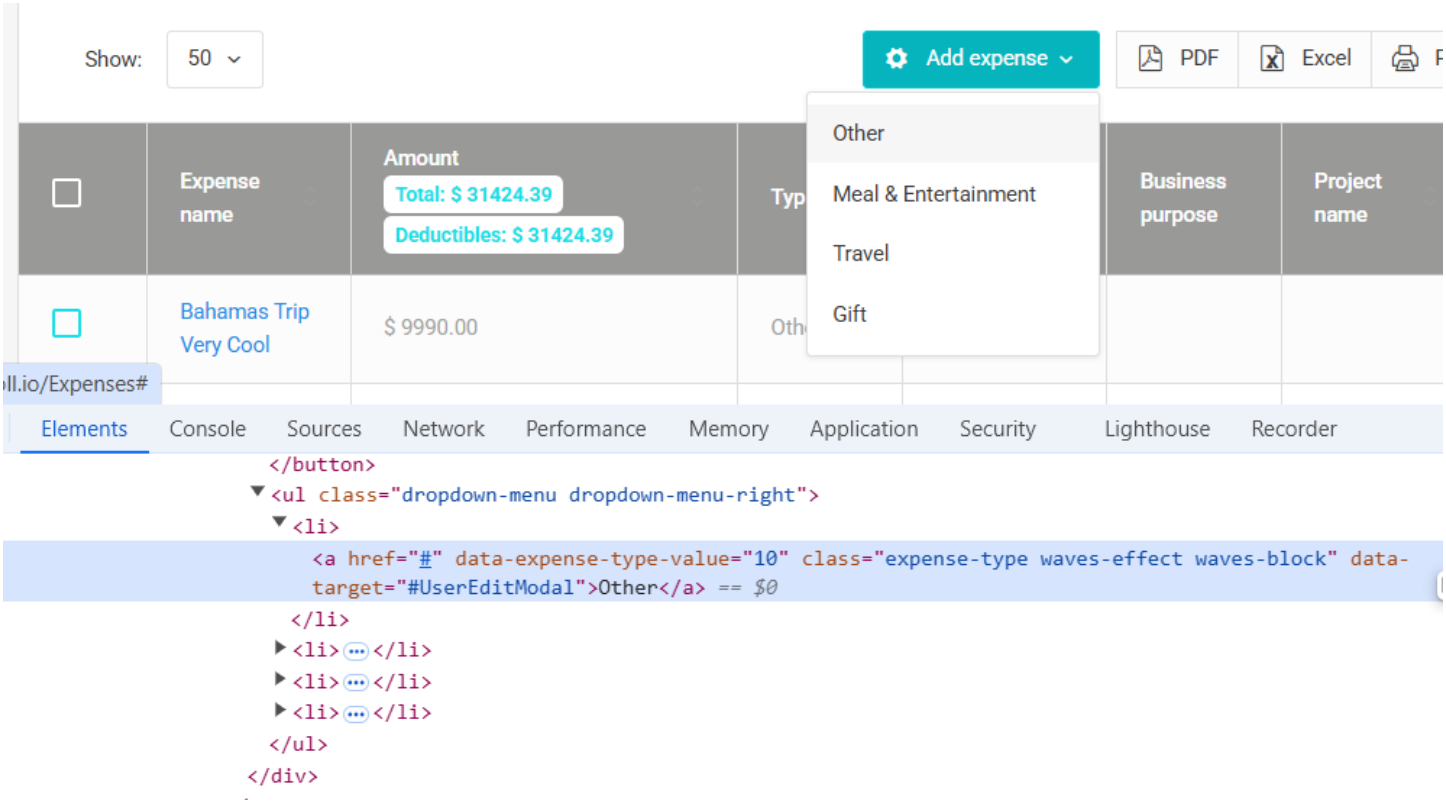
Dashboard Page: MealBDashboardPage



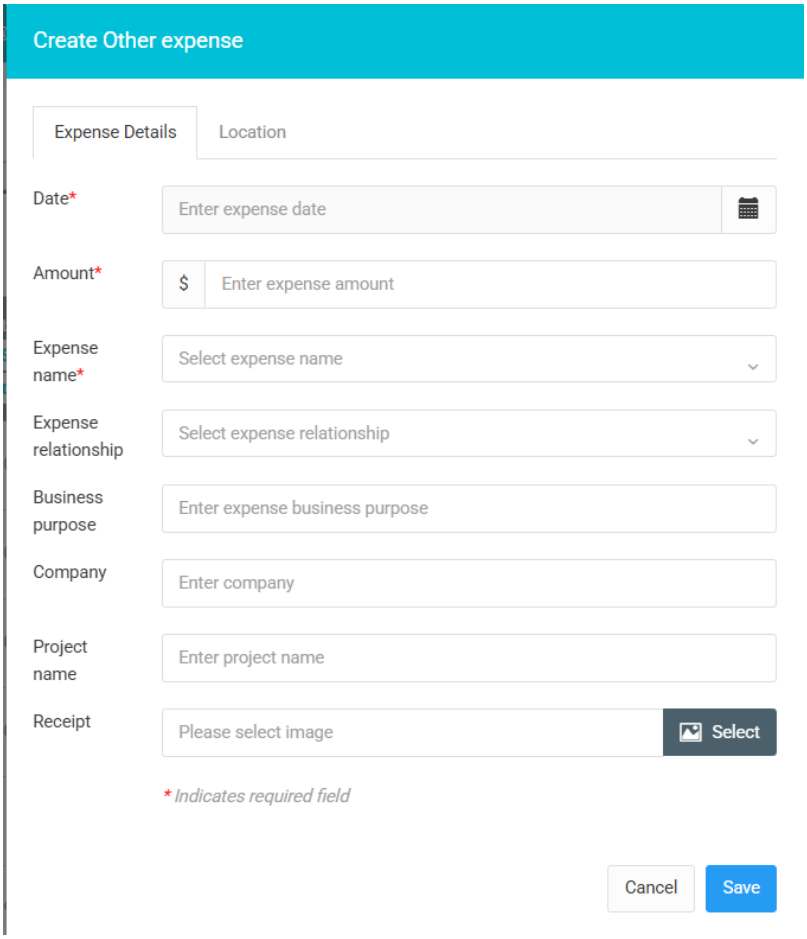


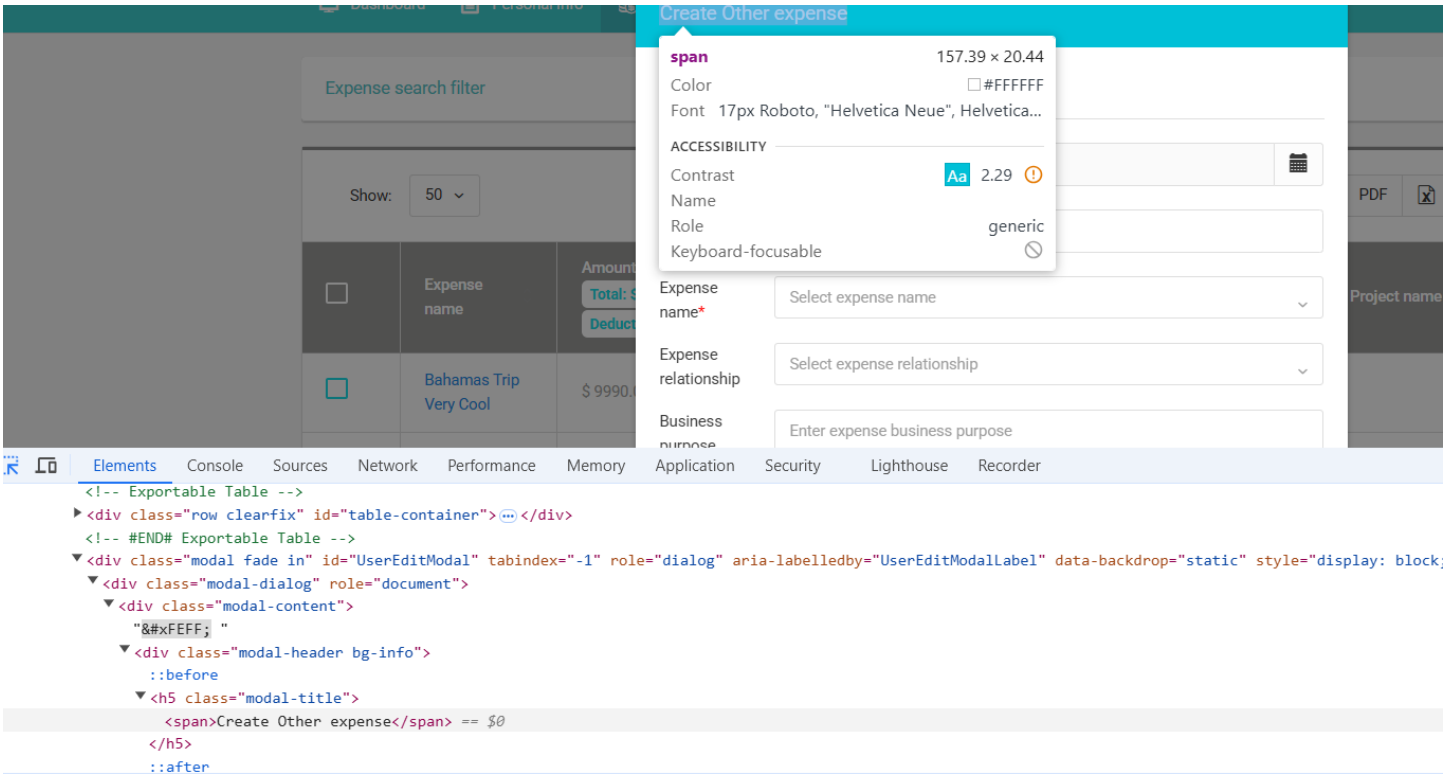
Expenses Page : MealBExpensesPage





Create Expenses Form Page : MealBCreateExpensesFormPage


Create Other expense
Expense Details
Location
Date*
Enter expense date
Amount*
$
Enter expense amount
Expense name*
Select expense name
Expense relationship
Select expense relationship
Business purpose
Enter expense business purpose
Company
Enter company
Project name
Enter project name
Receipt
Please select image
Select
* Indicates required field
Cancel
Save


Create Other expense
span
157.39 × 20.44
Color
#FFFFFF
Font
17px Roboto, "Helvetica Neue", Helvetica...
ACCESSIBILITY
Contrast
2.29
Name
Role
generic
Keyboard-focusable
Expense search filter
Show:
50
PDF
Expense name
Amount
Total:
Deduct
Project name
Bahamas Trip Very Cool
$ 9990.
Expense name*
Select expense name
Expense relationship
Select expense relationship
Business
Enter expense business purpose
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
Recorder
<!-- Exportable Table -->

<!-- #END# Exportable Table -->
<div class="modal fade in" id="UserEditModal" tabindex="-1" role="dialog" aria-labelledby="UserEditModalLabel" data-backdrop="static" style="display: block;

" "

::before
<h5 class="modal-title">
Create Other expense == $0
</h5>
::after

```sql
SELECT * FROM Expenses
WHERE CreatorUserId = (SELECT Id FROM AbpUsers WHERE UserName = 'user_test')
AND Name = 'Miami Trip'
AND Amount = 444.4
AND DeletionTime IS NULL
```

esults 1 ×

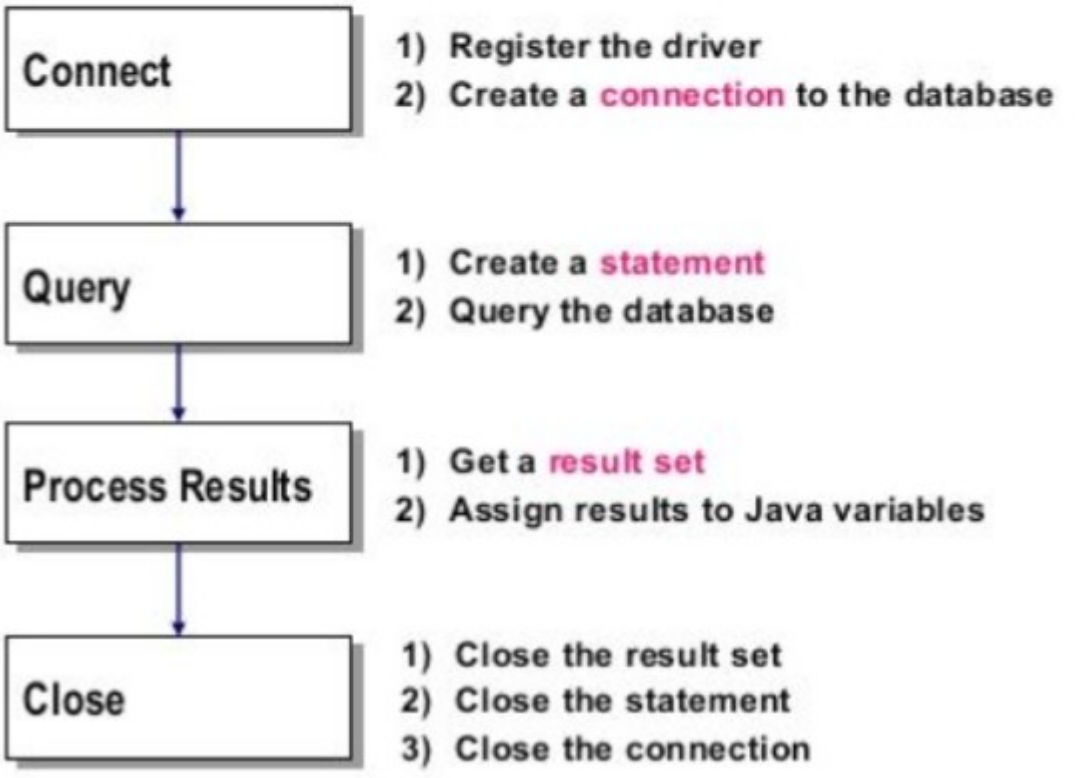
Enter a SQL expression to filter results (use Ctrl+Space)

Id	Amount	BusinessPurpose	CreationTime	CreatorUserId	DeleterUserId	DeletionTime	ExpenseDateTime	ExpenseRelationshipId
25,547	444.4	[NULL]	2024-01-28 18:01:34.973	12,572	[NULL]	[NULL]	2024-01-28 00:00:00.000	[NUL

DATABASE

- While testing we may need to compare some values we see in UI to backend (database)
- Below are most popular relational Databases used in market today:
  - MySQL
  - Microsoft SQL Server
  - Oracle
  - Postgres
- In selenium scripts we use **JDBC** library which helps to interact with database.
  - **JDBC** → Java Database Connector

JDBC CONNECTION STEPS (IQ)



SQL Server - Dependency

Add following dependency to POM.XML

```
<dependency>
    <groupId>com.microsoft.sqlserver</groupId>
    <artifactId>mssql-jdbc</artifactId>
    <version>8.2.2.jre8</version>
</dependency>
```

SQL Server- Steps to Connect

1. Load and Register the Database **Driver**
   a. Depending on which database you are using: MySQL, Oracle, Postgres etc.
   ```
   Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
   ```
2. Establish **Connection**.
   a.
   ```
   Connection con=DriverManager.getConnection("jdbc:sqlserver://34.235.0.4:1433;
   databaseName=MealBDb", "username", "password");
   ```
3. **Create Statement Object**
   a.
   ```
   Statement st= con.createStatement();
   ```
4. **Execute** the Statement
   a.
   ```
   ResultSet rs= st.executeQuery("Select * from dbo.Expenses");
   ```

SQL Server- Retrieving Data from DB

1. **Retrieve** the data from statement object

   To access the resultset object, we use a method called **next();**

```
while(resultset.next()) {
    expenseNameFromDb = resultset.getString("Name");

    System.out.println(expenseNameFromDb);
}
```

```
Feature: MealB Expenses
  Background:
    Given user navigates to MealB landing Page
    And clicks on Sign In button
    And logs in with employee account
  Scenario: Add Expense
    Given user is on the Dashboard page
    When user clicks on the Expenses tab
    And clicks on the Add Expense button
    And selects "Other" as the expense type
    Then a modal pop-up is shown as a form
    When user enters today date as the expense date
    And enters "155.00" as the expense amount
    And selects "Other" as expense name
    And enters "NY Trip" as other expense name
    And the user clicks the Save button
    And the expense "NY Trip" with amount "155.00" is listed on the Expenses page
    And a record for "NY Trip" with amount "155.00" is created in the database
```

```
package ui_automation.step_definitions;

import constants.Constants;
import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;
import org.junit.Assert;
import ui_automation.pages.*;
import ui_automation.utilities.ConfigurationReader;
import ui_automation.utilities.DBUtility;
import ui_automation.utilities.Driver;
import ui_automation.utilities.WaitHelper;

import java.sql.SQLException;
import java.util.List;
import java.util.Map;

public class MealBExpenseCommonSteps {
    private MealBLandingPage mealBLandingPage = new MealBLandingPage();
    private MealBLogingPage mealBLogingPage = new MealBLogingPage();
    private MealBDashboardPage mealBDashboardPage = new MealBDashboardPage();
    private MealBCreateExpensesFormPage mealBCreateExpensesFormPage = new
MealBCreateExpensesFormPage();
    private MealBExpensesPage mealBExpensesPage = new MealBExpensesPage();

    @Given("user navigates to MealB landing Page")
    public void user_navigates_to_MealB_landing_Page() {

Driver.getInstance().getDriver().get(ConfigurationReader.getProperty(Constants.UI_CONFIG,
Constants.MEALB_URL));
        WaitHelper.wait(2);


    }

    @Given("clicks on Sign In button")
    public void clicks_on_Sign_In_button() {
        mealBLandingPage.signInButton.click();
```

```
        WaitHelper.wait(2);
    }

    @Given("logs in with employee account")
    public void logs_in_with_employee_account() {
        mealBLogingPage.login(
                ConfigurationReader.getProperty(Constants.UI_CONFIG,
Constants.MEALB_USERNAME),
                ConfigurationReader.getProperty(Constants.UI_CONFIG,
Constants.MEALB_PASSWORD)
        );
        WaitHelper.shortWait();

    }

    @Given("user is on the Dashboard page")
    public void user_is_on_the_Dashboard_page() {
        String currentUrl = Driver.getInstance().getDriver().getCurrentUrl();
        WaitHelper.shortWait();
   //Assert.assertTrue("Dashboard verification
Failed !",currentUrl.endsWith(Constants.MEALB_DASHBOARD_ENDPOINT));
    }

    @When("user clicks on the Expenses tab")
    public void user_clicks_on_the_Expenses_tab() {
        mealBDashboardPage.expenseTab.click();
        WaitHelper.shortWait();
    }

    @When("clicks on the Add Expense button")
    public void clicks_on_the_Add_Expense_button() {
        mealBExpensesPage.addExpenseDropDown.click();
        WaitHelper.shortWait();
    }

    @When("selects {string} as the expense type")
    public void selects_as_the_expense_type(String expenseType) {
        if(expenseType.equals("Other")){
            mealBExpensesPage.otherExpenseOption.click();

        } WaitHelper.shortWait();

    }

    @Then("a modal pop-up is shown as a form")
    public void a_modal_pop_up_is_shown_as_a_form() {
        boolean
formIsDisplayed=mealBCreateExpensesFormPage.formModalHeader.isDisplayed();
        WaitHelper.shortWait();
        Assert.assertTrue("Form visibility verification Failed !",formIsDisplayed);

    }

    @When("user enters today date as the expense date")
    public void user_enters_todays_date_as_the_expense_date() {
        mealBCreateExpensesFormPage.expenseDateInputBox.click();
        WaitHelper.wait(1);
        mealBCreateExpensesFormPage.todaysDateButton.click();
        WaitHelper.wait(1);
```

```
    }

    @When("enters {string} as the expense amount")
    public void enters_as_the_expense_amount(String amount) {
        mealBCreateExpensesFormPage.expenseAmountInputBox.sendKeys(amount);
        WaitHelper.wait(2);

    }

    @When("selects {string} as expense name")
    public void selects_as_expense_name(String expenseName) {
        mealBCreateExpensesFormPage.expenseNameDropdown.click();
        WaitHelper.wait(2);
        mealBCreateExpensesFormPage.otherExpenseNameOption.click();
        WaitHelper.wait(2);

    }

    @When("enters {string} as other expense name")
    public void enters_as_other_expense_name(String expenseName) {
        mealBCreateExpensesFormPage.otherExpenseNameInputBox.sendKeys(expenseName);
        WaitHelper.wait(2);

    }

    @When("the user clicks the Save button")
    public void the_user_clicks_the_Save_button() {
        mealBCreateExpensesFormPage.saveButton.click();

    }

    @When("the expense {string} with amount {string} is listed on the Expenses page")
    public void the_expense_with_amount_is_listed_on_the_Expenses_page(String string,
String string2) {

    }

    @When("a record for {string} with amount {string} is created in the database")
    public void a_record_for_with_amount_is_created_in_the_database(String
expectedExpenseName, String expectedExpenseAmount) {
        // Open a connection to the database
        DBUtility.openConnection();

        // Define the SQL query to retrieve the expense record
        String query = "SELECT * FROM Expenses " +
                "WHERE CreatorUserId = (SELECT Id FROM AbpUsers WHERE UserName =
'user_test') " ;
//                + "AND Name = '" + expectedExpenseName + "'\n" +
//                "AND Amount = " + expectedExpenseAmount + "\n ";
              // + "AND DeletionTime IS NULL";

        //======

// Count first the amount of "Bahamas trip" records it should be 2
// Do the verification on the one that is not delelted
        List<Map<String, Object>> records = null;
        try {
            records = DBUtility.executeSQLQuery(query);
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
```

```
        for (Map<String, Object> record: records) {
            String expenseName = record.get("Name").toString();
            String expenseAmount = record.get("Amount").toString();
            if (expenseName.equals (expectedExpenseName) &&
record.get("IsDeleted").toString().equals("false")){
                // do some logic
                System.out.println("Amount is"+expenseAmount);
                System.out.println("Deleted"+record.get("IsDeleted").toString());
            }
                Assert.assertEquals (expectedExpenseAmount, expenseAmount);


        //=====
        // Execute the query and retrieve the results
//        List<Map<String, Object>> records = null;
//        try {
//            records = DBUtility.executeSQLQuery(query);
//        } catch (SQLException e) {
//            throw new RuntimeException(e);
//        }
//
//        // Verify that exactly one record is found
//        int actualAmountOfRecords = records.size();
//        int expectedAmountOfRecords = 1;
//        Assert.assertEquals("Expense DB record verification failed",
expectedAmountOfRecords, actualAmountOfRecords);
//
//        // Retrieve the first (and only) record from the results
//        Map<String, Object> record = records.get(0);
//
//        // Extract the 'Name' and 'Amount' fields from the record
//        String actualExpenseName = record.get("Name").toString();
//        String actualExpenseAmount = record.get("Amount").toString();
//
//        // Assert that the actual values match the expected values
//        Assert.assertEquals("Expense name does not match", expectedExpenseName,
actualExpenseName);
//        Assert.assertEquals("Expense amount does not match", expectedExpenseAmount,
actualExpenseAmount);
    }

}}


package runners;

import io.cucumber.junit.Cucumber;
import io.cucumber.junit.CucumberOptions;
import org.junit.runner.RunWith;

@RunWith(Cucumber.class)
@CucumberOptions(
        plugin = {"pretty","html:target/cucumber", "json:target/cucumber.json"},
        features="src/test/resources/uiFeatures",
        glue="ui_automation.step_definitions",
        tags="@OMBExpenses",
        dryRun = false,
        monochrome = true
)

public class UITestRunner {
```

```
}
```

```
package ui_automation.pages;

import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;
import ui_automation.utilities.Driver;
import ui_automation.utilities.WaitHelper;

public class MealBLogingPage {
    public MealBLogingPage() {
        PageFactory.initElements(Driver.getInstance().getDriver(), this);
    }

    @FindBy(css = "[name=usernameOrEmailAddress]")
    public WebElement usernameInputBox;

    @FindBy(css = "[name=Password]")
    public WebElement passwordInputBox;

    @FindBy(id = "LoginButton")
    public WebElement loginButton;

    public void login(String username, String password){

        usernameInputBox.sendKeys(username);
        WaitHelper.shortWait();
        passwordInputBox.sendKeys(password);
        WaitHelper.shortWait();
        loginButton.click();
        WaitHelper.shortWait();
    }
}
```

```
package ui_automation.pages;

import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;
import ui_automation.utilities.Driver;

public class MealBExpensesPage {
    public MealBExpensesPage() {
        PageFactory.initElements(Driver.getInstance().getDriver(), this);
    }
    @FindBy(xpath = "//span//button[contains(text(),'Add expense')]")
    public WebElement addExpenseDropDown;

    @FindBy(xpath = "//div[@class=\"panel panel-flat\"]//a[text()='Other']")
    public WebElement otherExpenseOption;


}
```

```
package ui_automation.pages;

import org.openqa.selenium.WebElement;
```

```java
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;
import ui_automation.utilities.Driver;

public class MealBLandingPage {
    public MealBLandingPage() {
        PageFactory.initElements(Driver.getInstance().getDriver(), this);
    }

    @FindBy(xpath = "//*[text()='SIGN IN']")
    public WebElement signInButton;
}
```

```java
package ui_automation.pages;

import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;
import ui_automation.utilities.Driver;

public class MealBDashboardPage {
    public MealBDashboardPage() {
        PageFactory.initElements(Driver.getInstance().getDriver(), this);
    }

    @FindBy(xpath = "//*[contains(text(),'Expenses')]")
    public WebElement expenseTab;
}
```

```java
package ui_automation.pages;

import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;
import ui_automation.utilities.Driver;

public class MealBCreateExpensesFormPage {
    public MealBCreateExpensesFormPage() {
        PageFactory.initElements(Driver.getInstance().getDriver(), this);
    }
    @FindBy(xpath = "//*[text()='Create Other expense']")
    public WebElement formModalHeader;

    @FindBy(id = "ExpenseDateTime")
    public WebElement expenseDateInputBox;
    @FindBy(css = "[class=picker__button--today][aria-controls=ExpenseDateTime]")
    public WebElement todaysDateButton;
    @FindBy(id = "Amount")
    public WebElement expenseAmountInputBox;

    @FindBy(id = "name")
    public WebElement otherExpenseNameInputBox;

    @FindBy(css = "[data-id=OtherExpenseNameId]")
    public WebElement expenseNameDropdown;

    @FindBy(xpath="//li[@data-original-index='4']//*[text() = 'Other']")
    public WebElement otherExpenseNameOption;
    @FindBy(xpath="//*[text() = 'Save']")
```

```
    public WebElement saveButton;
}
```

Task: Orkhan Class

```
@regression @mbExpenses
Feature: This feature file validates MealB Expenses Functionality
Scenario: User should be able to create Meal & Entertainment Expense on MealB
Given user navigates to MealB Login Page
When user logs in to MealB with valid credentials
Then user navigates to "Expenses" tab on MealB
Then user navigates to "Create Meal & Entertainment" expense modal window on MealB
Then user completes all mandatory fields on Create Meal & Entertainment expense modal
window
Then user clicks on save button on Create Meal & Entertainment expense modal window
And user validates created Meal & Entertainment expense on Expenses Page
@run
Scenario: User should be able to compare UI expenses with DB Expenses
Given user navigates to MealB Login Page
When user logs in to MealB with valid credentials
Then user navigates to "Expenses" tab on MealB
And user validates UI expenses with DB expenses
```

```
package ui_automation.step_definitions;
import io.cucumber.java.en.And;
import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;
import org.junit.Assert;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import ui_automation.pages.MBExpensesPage;
import ui_automation.utilities.*;
import java.sql.SQLException;
import java.util.List;
import java.util.Map;
public class MBExpensesPageSteps {
WebDriver driver = Driver.getInstance().getDriver();
MBExpensesPage mbExpensesPage = new MBExpensesPage(driver);
String todaysDate;
double amount;
String expenseName;
@Given("user navigates to MealB Login Page")
public void user_navigates_to_MealB_Login_Page() throws InterruptedException {
Thread.sleep(2000);
String url = ConfigurationReader.getProperty("ui-config.properties", "mealb.url");
driver.get(url);
}
@When("user logs in to MealB with valid credentials")
public void user_logs_in_to_MealB_with_valid_credentials() throws InterruptedException {
Thread.sleep(2000);
String username = ConfigurationReader.getProperty("ui-config.properties",
"mealb.username");
String password = ConfigurationReader.getProperty("ui-config.properties",
"mealb.password");
driver.findElement(By.name("usernameOrEmailAddress")).sendKeys(username);
```

```
driver.findElement(By.name("Password")).sendKeys(password);
driver.findElement(By.id("LoginButton")).click();
}
@Then("user navigates to {string} tab on MealB")
public void user_navigates_to_tab_on_MealB(String tab) throws InterruptedException {
Thread.sleep(2000);
switch (tab) {
case "Expenses":
driver.findElement(By.xpath("//*[contains(text(),'Expenses')]")).click();
break;
}
}
@Then("user navigates to {string} expense modal window on MealB")
public void user_navigates_to_expense_modal_window_on_MealB(String expenseType) throws
InterruptedException {
Thread.sleep(2000);
mbExpensesPage.addExpenseDropdown.click();
Thread.sleep(1000);
switch (expenseType) {
case "Create Meal & Entertainment":
mbExpensesPage.mealAndEntertainmentOption.click();
break;
}
}
@Then("user completes all mandatory fields on Create Meal & Entertainment expense modal
window")
public void
user_completes_all_mandatory_fields_on_Create_Meal_Entertainment_expense_modal_window()
throws Exception {
Thread.sleep(2000);
/* establishing the connection to the excel file */
String filePath = System.getProperty("user.dir") + "\\src\\test\\resources\\testData\\
testData.xlsx";
ExcelUtility.setExcelFile(filePath, "Sheet1");
/* retrieving data from excel file */
todaysDate = DateHelper.getTodaysDate("MM/dd/yyyy");
amount = ExcelUtility.getCellDataAsDouble(7, 3);
expenseName = ExcelUtility.getCellDataAsString(7, 0);
/* complete mandatory fields */
mbExpensesPage.dateField.click();
Thread.sleep(1000);
driver.findElement(By.xpath("(//*[@aria-label='"+todaysDate+"'])[3]")).click();
mbExpensesPage.amountTextBox.sendKeys(String.valueOf(amount));
mbExpensesPage.expenseNameTextBox.sendKeys(expenseName);
}
@Then("user clicks on save button on Create Meal & Entertainment expense modal window")
public void
user_clicks_on_save_button_on_Create_Meal_Entertainment_expense_modal_window() throws
InterruptedException {
Thread.sleep(2000);
mbExpensesPage.saveButton.click();
}
@Then("user validates created Meal & Entertainment expense on Expenses Page")
public void user_validates_created_Meal_Entertainment_expense_on_Expenses_Page() throws
InterruptedException {
Thread.sleep(5000);
List<WebElement> allExpenses = driver.findElements(By.xpath("//table[@id='expenses-
table']/tbody/tr/td[2]"));
boolean isNewExpenseDisplayed = false;
for (WebElement expense : allExpenses) {
String actualExpenseNameValue = expense.getText();
```

```
if (actualExpenseNameValue.equals(expenseName)) {
isNewExpenseDisplayed = true;
break;
}
}
Assert.assertTrue("New Expense validation failed!", isNewExpenseDisplayed);
}
@And("user validates UI expenses with DB expenses")
public void userValidatesUIExpensesWithDBExpenses() throws InterruptedException,
SQLException {
Thread.sleep(5000);
/* create a connection to MealB DB */
DBUtility.openConnection();
/* establish the statement and execute the query */
String query = "SELECT Name from Expenses as expenses\n" +
"join AbpUserAccounts as users\n" +
"on users.UserId = expenses.CreatorUserId\n" +
"WHERE users.UserName = 'walmart' and expenses.DeletionTime is NULL\n" +
"ORDER by expenses.Name ASC;";
List<Map<String, Object>> dataFromDB = DBUtility.executeSQLQuery(query);
int countFromUI;
int countFromDB = dataFromDB.size();
/* extract count value from UI and compare with DB */
String expenseCountMessage = mbExpensesPage.expensesTableFooter.getText();
String[] words = expenseCountMessage.split(" ");
countFromUI = Integer.parseInt(words[5]);
Assert.assertEquals("Expense count verification failed!", countFromDB, countFromUI);
/* click on Expense Name column to sort in alph ASC order */
Thread.sleep(2000);
driver.findElement(By.xpath("//table[@id='expenses-table']/thead//th[2]")).click();
/* validate DB expenses with UI expenses one by one */
Thread.sleep(3000);
int i = 0;
List<WebElement> allExpenses = driver.findElements(By.xpath("//table[@id='expenses-
table']/tbody/tr/td[2]"));
for (Map<String, Object> data : dataFromDB) {
String expectedDataFromDB = data.get("Name").toString();
String actualDataFromUI = allExpenses.get(i).getText();
Assert.assertEquals("Data verification failed!", expectedDataFromDB, actualDataFromUI);
i++;
}
}
}
```

```
package ui_automation.pages;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;
public class MBExpensesPage {
WebDriver driver;
public MBExpensesPage (WebDriver driver) {
this.driver = driver;
PageFactory.initElements(driver, this);
}
@FindBy(xpath = "(//*[contains(@class,'add-expense-button')])[2]")
public WebElement addExpenseDropdown;
@FindBy(xpath = "(//*[text()='Meal & Entertainment'])[4]")
public WebElement mealAndEntertainmentOption;
@FindBy(id = "ExpenseDateTime")
```

```
public WebElement dateField;
@FindBy(id = "Amount")
public WebElement amountTextBox;
@FindBy(id = "name")
public WebElement expenseNameTextBox;
@FindBy(css = ".save-button")
public WebElement saveButton;
@FindBy(id = "expenses-table_info")
public WebElement expensesTableFooter;
}
```